Année : **2007**

THESE

présentée à

L'UFR des Sciences et Techniques
de l'Université de Franche-Comté

pour obtenir le

GRADE DE DOCTEUR DE L'UNIVERSITE DE FRANCHE-COMTE

en Automatique
(Ecole Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques)

Vers une conception conjointe des architectures du produit et de l'organisation du projet dans le cadre de l'Ingénierie Système

par

Ghassen HARMEL

Soutenue le **05 juillet 2007** devant la Commission d'examen :

Rapporteurs :
M. Alain BERNARD, Professeur, Ecole Centrale de Nantes, IRCCyN
M. Gérard MOREL, Professeur, UHP, CRAN

Examinateurs :
M. Michel ALDANONDO, Professeur, Ecole des Mines d'Albi-Carmaux, CGI
M. Jean-François BOUJUT, Professeur, INP de Grenoble, Laboratoire G-SCOP
M. Dominique LOISE, Responsable AQO Conception Système GMP, PSA
M. Jean-Pierre MICAELLI, Maître de Conférences, UTBM, RECITS

Directeurs de thèse :
Mme. Maryvonne DULMET, Maître de Conférences - HDR, UFC, LAB
M. Eric BONJOUR, Maître de Conférences, UFC, LAB

REMERCIEMENTS

Les travaux présentés dans ce mémoire ont été réalisés au sein du Laboratoire d'Automatique de Besançon (LAB). Je tiens donc à remercier Monsieur Alain Bourjault, Directeur du LAB et son successeur Monsieur Nicolas Chaillet, de m'avoir offert les conditions favorables pour effectuer ces travaux de recherche.

Ces travaux ont été dirigés par Madame Maryvonne Dulmet, Maître de Conférences (HDR) à l'Université de Franche-Comté, et de Monsieur Eric Bonjour, Maître de Conférences à l'Université de Franche-Comté. Je tiens vivement à leur adresser ma reconnaissance pour la confiance et la liberté qu'ils ont su m'accorder ainsi que pour leurs conseils et leur disponibilité.

Je tiens à remercier Monsieur Alain Bernard, Professeur à l'Ecole Centrale de Nantes, ainsi que Monsieur Gérard Morel, Professeur à l'Université Henri Poincaré pour m'avoir fait l'honneur d'être les rapporteurs de ce mémoire et pour m'avoir apporté de précieuses remarques pour améliorer la qualité de ce document.

Je tiens également à remercier Monsieur Michel Aldanondo, Professeur à l'Ecole des Mines d'Albi-Camaux, d'avoir accepté de présider ma soutenance de thèse.

Je souhaite aussi remercier les autres membres du jury, Monsieur Jean-François Boujut, Professeur à l'Institut National Polytechnique de Grenoble, Monsieur Jean-Pierre Micaëlli, Maître de Conférences à l'Université de Technologie Belfort-Montbéliard et Monsieur Dominique Loise, Responsable Assurance Qualité Opérationnelle conception système GMP au sein de PSA Peugeot-Citroën, pour m'avoir conseillé durant ma thèse et pour avoir participé au jury de ma soutenance.

J'adresse particulièrement mes remerciements à Monsieur Norbert Lartigue, Directeur de la Direction des Plateformes Métiers Techniques Organes et l'ensemble des directeurs métiers et ingénieurs que nous avons rencontrés sur le site de PSA Peugeot-Citroën à La Garenne-Colombes et qui nous ont permis de valider notre travail sur des projets de développement de moteurs et de boites de vitesse.

TABLE DES MATIERES

TABLE DES ILLUSTRATIONS

Figures

Tableaux

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Depuis plusieurs années, de nombreux chercheurs et industriels notent que la complexité des produits ne cesse d'augmenter, pour satisfaire au mieux les exigences croissantes provenant des différents acteurs du cycle de vie du produit (dont les clients, les organismes de normalisation, le service maintenance …). Pour maîtriser cette complexité et améliorer les performances en conception, de nombreux modèles, méthodes et outils ont été développés pour aider les concepteurs dans leurs activités. Les outils de calcul et de simulation prennent une place de plus en plus importante pour accroître la compétitivité des produits industriels, sur les trois critères : Qualité, Coûts et Délai de mise sur le marché.

Les organisations industrielles ont évolué afin de répondre à ces impératifs. Aux organisations séquentielles, cloisonnées par une hiérarchie professionnelle et culturelle, ont succédé les plateaux projets, l'ingénierie concourante ou simultanée, le travail collaboratif et le co-développement avec des partenaires extérieurs, modifiant ainsi les frontières du processus de conception. Ces nouvelles organisations permettent de mettre en œuvre des processus de conception plus courts par la parallélisation des tâches, aboutissant à des produits innovants de meilleure qualité et moins chers, tout en intégrant les acteurs de tout le cycle de vie du produit (de la conception à son démantèlement) au plus tôt dans le processus de conception. Depuis quelques années, les concepteurs doivent aussi faire face au défi du développement durable. Il s'agit d'intégrer des valeurs et des contraintes environnementales et sociales.

Offrir aux concepteurs des modèles, méthodes et outils qui supportent de façon efficace et efficiente leur travail dans un contexte de conception intégrée relève donc d'une problématique stratégique pour les entreprises.

Cependant, un travail de thèse ne saurait prétendre y répondre dans sa globalité. C'est pourquoi nous préciserons le sujet et les enjeux de notre travail de recherche. Nos travaux s'appuient sur des relations avec des entreprises qui mettent en œuvre la conception modulaire. Cette notion de modularité est liée au besoin de diversification des produits et à la conception de familles de produits. En effet, des combinaisons et architectures appropriées de modules peuvent créer potentiellement un grand nombre de produits différents du point de vue du client, appartenant à ce qu'on appelle une famille de produits. Cette approche donne la possibilité aux concepteurs de raisonner en termes de famille de produits avec des modules communs pour limiter les coûts et des modules

distinctifs pour personnaliser le produit selon les attentes des clients. La conception modulaire dans le contexte de l'Ingénierie Système introduit une étape importante dans le processus de conception, à savoir la conception de l'architecture du produit. Se réalisant avant la conception détaillée des constituants du produit, la conception de l'architecture, passant par une vue fonctionnelle et une vue organique, permet de spécifier les exigences pour chaque constituant et de figer progressivement leurs interfaces physiques et ainsi de surmonter la complexité liée au produit.

Ces besoins industriels ont été approfondis dans le cadre d'un projet de recherche, impliquant d'une part, des chercheurs du LAB et de RECITS[1], d'autre part, une Direction du Groupe PSA Peugeot Citroën, chargée de la conception des GMP et des LAS[2]. Notre travail a nécessité de nombreux entretiens[3] avec des architectes du système GMP, pour comprendre leurs activités et leurs besoins. Nous avons recueilli des données dans le cadre de projets de développement d'un nouveau moteur et de différentes boîtes de vitesse. Les architectes ont ensuite collaboré lors de l'interprétation de nos résultats de simulation.

D'une façon générale, lorsqu'une entreprise prend la décision stratégique de lancer une nouvelle famille de produits ou de reconcevoir un produit existant (par exemple, dans l'automobile, des moteurs diesel ou hybride pour répondre à un durcissement des normes anti-pollution, Euro V), l'architecte système doit concevoir ou faire évoluer l'architecture. Mais il joue aussi le rôle de chef de projet et doit concevoir ou faire évoluer en même temps, l'organisation du projet (nouveau découpage du projet en équipes) pour la rendre plus performante.

Notre problématique a consisté à développer des modèles et méthodes permettant d'aider les architectes système dans cette double activité. Ce contexte se retrouve dans toutes les entreprises dont la complexité des produits nécessite des architectures complexes du produit et de l'organisation de projet (automobile, aéronautique, télécom, machines spéciales, construction …). En résumé, nous proposons dans ce travail une méthode de conception conjointe des architectures du produit et de l'organisation du projet dans les phases préliminaires du processus de conception, où le rôle de l'architecte est déterminant.

Ce mémoire décrit la méthodologie proposée et son application à des exemples industriels. Il se compose de cinq chapitres :

Le premier chapitre pose le cadre de notre travail : l'Ingénierie Système appliquée au produit et au projet. Nous introduisons les principaux concepts de la conception modulaire (méthodes de modularisation, métriques de modularité, …) et de l'architecture du produit (définitions, typologies, …) que nous étendons au projet pour définir l'architecture de l'organisation du

[1] RECITS, Laboratoire de recherche sur les choix industriels, technologiques et scientifiques, EA 3897, UTBM, Sévenans (90)

[2] Le Groupe Moto-Propulseur (GMP) correspond principalement au moteur, à la boîte de vitesse et à la transmission. La Liaison Au Sol (LAS) correspond au châssis : essieux avant et arrière, systèmes de direction, de freinage, de suspension ...

[3] A titre personnel, j'ai participé à plus de 40 entretiens et j'ai passé environ 15 jours sur le site de PSA à La Garenne Colombes.

projet. Ce chapitre nous permet alors de converger vers notre vision de la modélisation de l'architecture du produit et de l'organisation de conception.

Le deuxième chapitre présente l'outil de modélisation des architectures que nous avons adopté dans notre travail : les matrices de couplage. Cet outil permet à la fois de formaliser les interactions qui lient les éléments du système et de mettre en œuvre une méthode de modularisation.

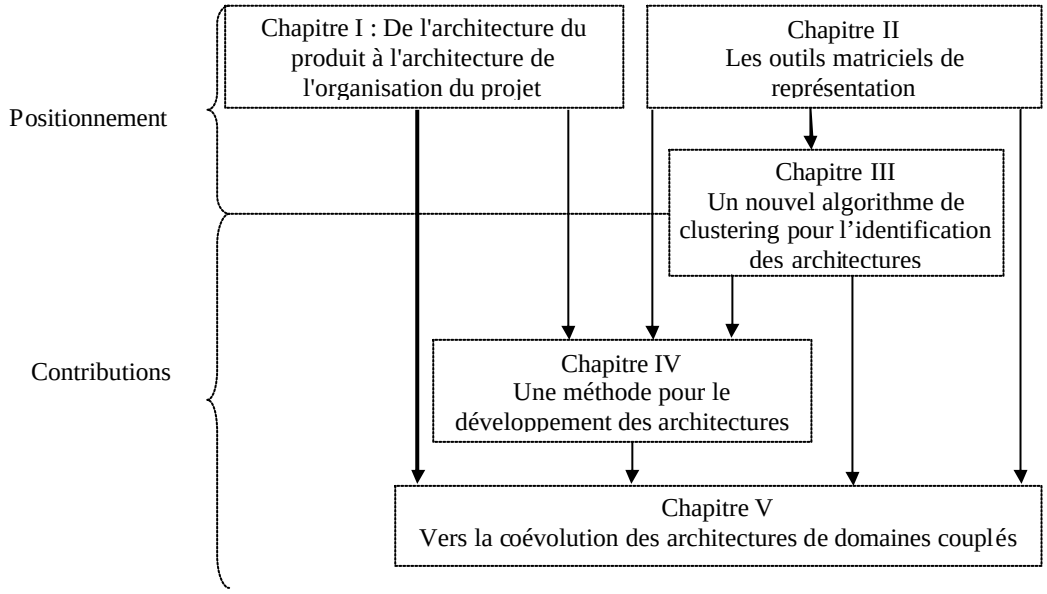
Dans le chapitre 3, nous décrivons un algorithme de clustering que nous avons utilisé comme référence et qui supporte une méthode de modularisation. Nous exposons ensuite l'algorithme que nous avons développé, ses caractéristiques et ses avantages.

Dans le chapitre 4, nous présentons quatre situations de conception des architectures des domaines du produit. Ces situations dépendent du type de données disponibles à un moment du processus de conception. Nous proposons alors, pour chaque situation, une méthode de conception des architectures, faisant appel à un traitement flou et/ou à des opérations matricielles. Chacune de ces situations est ensuite illustrée par une application à la conception d'un moteur thermique dans l'industrie automobile. La démarche présentée dans ce chapitre est une vision statique de la conception des architectures.

Puis nous montrons, dans le chapitre 5, la nécessité de faire « coévoluer » les architectures couplées. Nous proposons l'exploration des incertitudes comme méthode pour suivre l'évolution des systèmes (perturbations) étudiés. Nous développons une méthode basée sur un traitement flou pour faire coévoluer les architectures perturbées et pour les rendre cohérentes.

La conclusion générale de ce mémoire résume nos contributions, tout en soulignant leurs limites et les perspectives qu'elles ouvrent.

Le schéma suivant montre la structure globale de ce mémoire.



CHAPITRE I

CHAPITRE I

DE L'ARCHITECTURE DU PRODUIT A L'ARCHITECTURE DE L'ORGANISATION DU PROJET

Les évolutions technologiques, les contraintes économiques et la concurrence mondiale de plus en plus rude poussent les entreprises à chercher de nouvelles solutions pour rester performantes et maintenir leur avantage sur les marchés. Pour cela, elles doivent sans cesse développer de nouveaux produits innovants, avec une meilleure qualité, avec des coûts de production maîtrisés et des délais réduits de mise sur le marché.

Le domaine de la conception a été exploré par de nombreux chercheurs pour développer de nouveaux outils et méthodes pouvant aider les concepteurs à améliorer leurs performances. Des méthodologies voire une science de la conception [Perrin, 2001], [Forest et al., 2005] ont été développées, supportées par des concepts, des modèles, des règles et des normes, faisant appel à des domaines disciplinaires variés.

Le pilotage des activités de conception a pris une place croissante avec d'une part, l'arrivée de méthodes de planification de grands projets dès les années 60 et d'autre part, avec la mise en place d'organisations de conception plus cohérentes avec les nouveaux objectifs des projets de développement de nouveaux produits (par exemple, organisations matricielles et par projet).

Les travaux de recherche concernant **la conception de produits mécatroniques** ont pour objet d'expliciter et de développer des méthodes et outils d'aide à la conception en intégrant des exigences sur tout le cycle de vie du produit, depuis les phases d'invention jusqu'à celles de production et de maintenance. Ces travaux sont intimement liés à la fois aux problématiques organisationnelles et stratégiques de l'entreprise.

Ce premier chapitre vise à présenter la base et le cadre global du travail présenté dans ce mémoire. C'est pourquoi nous débuterons par la présentation de quelques processus et démarches de conception en nous attardant sur l'Ingénierie Système. Ensuite, nous

introduirons le concept d'architecture tel qu'il est utilisé pour le produit et ensuite, nous l'étendrons à l'organisation du projet.

1. Les processus et démarches de conception

La conception est une activité complexe, nécessitant l'intégration de multiples points de vue (cognitif, technique, social, économique, organisationnel,...), chaque point de vue fournissant un ensemble d'exigences et de contraintes [Pahl et Beitz, 1996], qui sont évolutives et souvent mal connues au début du projet. Les travaux portant sur cette activité couvrent de nombreux aspects, depuis les théories de la conception [Hatchuel et Weil, 2002], [Micaëlli et Forest, 2003], [Perrin, 2001], jusqu'à la modélisation du processus de conception en passant par la capitalisation des connaissances produites au cours des activités de conception [Harani, 1997], [Eynard, 1999], [Bernard, 2000], [Menand, 2002]. Certains travaux concernent l'organisation de la conception [Midler, 1998], le pilotage des processus [Perrin, 1999], et l'intégration des compétences et des métiers dans le processus de conception [Bouchikhi, 1990], [Lefebvre et al., 2002]. D'autres travaux ont pour but de développer des supports pour aider les concepteurs dans leur travail collaboratif [Robin et al., 2004]. En effet, une conséquence de la complexité de cette activité est que celle-ci n'est jamais individuelle mais présente un caractère collectif et distribué [Blanco, 1998]. Les acteurs de la conception proviennent d'univers disciplinaires et culturels différents, d'où le rôle déterminant des mécanismes de coopération et de compréhension mutuelle.

La norme AFNOR X50-127 définit le processus de conception de la façon suivante : « partant des besoins exprimés, le processus de conception définit pas à pas le produit qui doit répondre aux besoins et aux attentes par des choix successifs portant sur des points de plus en plus détaillés ». La tâche du concepteur revient donc à transformer un besoin client, exprimé en termes de fonctions, en une description détaillée et suffisamment complète du produit. La description finale du produit est généralement représentée sous diverses formes : physique (maquette), graphique (plan), numérique ou simplement textuelle.

De nombreux modèles de processus de conception, plus ou moins détaillés, existent dans la littérature de l'ingénierie de la conception [Ulrich and Eppinger, 2000; Roozenburg et Eekels, 1995; Otto et Wood, 2001 ; EIA 632 ; ISO 15288]. Nous nous intéresserons essentiellement aux modèles développés par l'association des ingénieurs allemands [VDI 2221 ; VDI 2206], en particulier, par Pahl et Beitz [1996] ainsi qu'aux modèles provenant d'une norme sur l'ingénierie système qui se focalise sur le processus de développement [IEEE 1220].

1.1. Un modèle du processus de conception

Dans leur ouvrage, Pahl et Beitz [1996] ont proposé un modèle théorique structurant le déroulement progressif du processus de conception d'un produit. Ce modèle simplifié comporte quatre phases et a prouvé son efficacité puisqu'il a été pris comme base de référence

par différents travaux de recherches en conception. Par la suite, chaque entreprise a apporté ses propres modifications selon les spécificités de ses activités et de son produit.

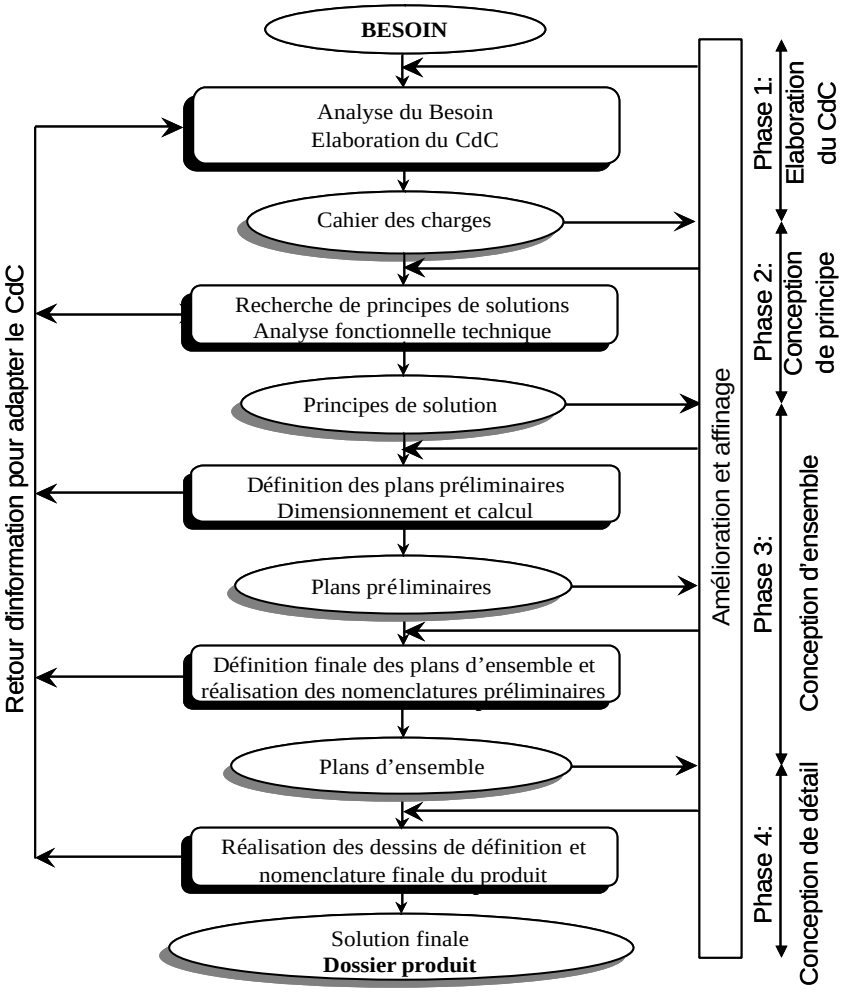


Figure I-1. Modèle du processus de conception [Pahl et Beitz, 1996]

Comme le montre la figure I-1, le modèle inclut quatre phases itératives :

- L'analyse du besoin (*Product planning and clarifying the task*) permet de ré-exprimer ce dernier en un langage technique, compréhensible par l'ensemble des acteurs de conception. Le cahier des charges (CdC), résultat de cette phase, regroupe toutes les données fonctionnelles voulues du produit.

- La conception de principe (*Conceptual design*) vise à spécifier l'ensemble des principes technologiques et solutions envisageables pour répondre au besoin. Cette phase est complétée par une analyse fonctionnelle technique puis une évaluation de toutes les solutions et les principes trouvés afin de ne retenir que les meilleurs.

- La conception d'ensemble (*Embodiment design*) s'intéresse à concrétiser la solution retenue en commençant par mettre au point les grands axes de la solution adoptée (plans préliminaires) puis, par la suite, améliorer progressivement cette solution. Il s'agit de

donner la structure générale, de faire les calculs et dimensionnements de base puis de regrouper les plans d'ensemble.

- La conception de détail (*Detail design*) consiste à réaliser les dernières modifications sur la structure du produit, à déterminer et dimensionner tous ses composants ainsi que toutes les liaisons qui les relient et enfin à définir les moyens et les modes de production. A la fin de cette étape, le dossier produit est finalisé. Des tests de fiabilité par expérimentations sur prototype peuvent être effectués en complément.

Ce modèle du processus de conception est important pour expliquer les bases d'une démarche « systématique » de conception d'un produit mécanique. Cependant, comme tout modèle, il ne peut pas tout représenter : il ne met pas l'accent sur la décomposition systémique ou sur la définition progressive (éventuellement, avec des rangs de maturité) des produits complexes et/ou innovants.

En Allemagne, Pahl et Beitz ont fait partie de différents groupes d'ingénieurs allemands (*VDI - Verein Deutscher Ingenieure* : Association d'ingénieurs allemands) élaborant des directives (*Richtlinie*) portant sur des méthodes de développement de produits. Très productifs, ces groupes ont proposé différents modèles du processus de conception, en particulier, selon le type de produits [VDI 2221 ; VDI 2206].

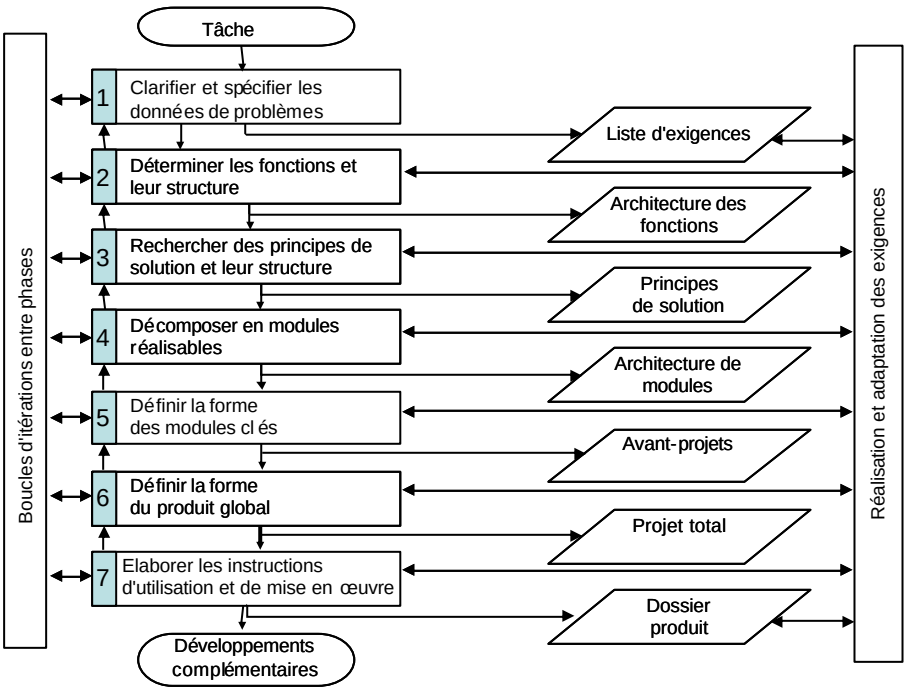


Figure I-2. Processus de développement de produits (d'après VDI 2221)

Un autre modèle du processus de développement de produit, moins connu que le précédent [Pahl et Beitz, 1996], est très intéressant pour décrire les tâches critiques en conception

(figure I-2). Il permet de situer l'élaboration de l'« architecture[4] fonctionnelle » et de l'« architecture de modules »[5] et leur amélioration, leur raffinement dans les phases préliminaires de conception de principe (phases 1, 2, 3) et de conception d'ensemble (phases 4, 5, 6, 7).

Dans la suite de cette partie, nous nous intéresserons à l'ingénierie intégrée, et dans la partie 3, nous nous focaliserons sur l'Ingénierie Système. Ces approches, en partie fondées sur les modèles précédents et adaptées au développement de produits mécatroniques complexes, ont vu le jour au début des années 90 et ont été mises en œuvre dans des entreprises de différentes tailles.

1.2. Une démarche de conception : l'ingénierie intégrée[6]

L'ingénierie intégrée (ou Concurrent Engineering, notée par la suite CE) [Kusiak et Wang, 1993], [Lawson et Karandikar, 1994], [Prasad, 1997], [Midler, 1998] dite aussi ingénierie simultanée ou concourante ou encore congruente [Trassaert, 2002] est apparue au milieu des années 80 comme une nouvelle forme d'organisation en conception. Elle répond à un besoin d'amélioration de la compétitivité des entreprises, qui doivent développer leur produit et son système de production, toujours plus rapidement, moins cher et avec une assurance de qualité. Deux grands principes sont mis en œuvre : la simultanéité et l'intégration. Le premier consiste à réaliser en même temps différentes activités concourant à la conception du produit et de son système de production, le second est caractérisé par l'établissement d'une interdépendance entre les différentes phases du projet, par la prise en compte, à chaque phase du développement, des considérations relatives à l'ensemble du cycle de vie du produit, depuis sa conception jusqu'à sa mise à disposition (coût, qualité, délai, besoins du consommateur…). Quand la conception du produit atteint un certain stade, des informations préliminaires sont transmises aux concepteurs du système de production et la conception commence dès que possible. Finalement, le projet de conception d'un produit complexe (tel un véhicule) est lui-même complexe car son organisation se construit graduellement en fonction des informations et des contraintes très variées, évolutives et incertaines qui sont traitées progressivement par les acteurs du projet. De ce fait, le CE est un défi managérial et organisationnel.

[4] Dans une première approche, retenons que l'architecture est un terme, issu du latin, qui désigne une discipline qui associe art et science de construire des bâtiments terrestres ou navals et des structures. Par extension, le terme d'architecture est également utilisé pour désigner la conception ou l'acte de concevoir des systèmes complexes. Dans ce cas, on fait référence à la structure générale du système.

[5] Les termes originaux sont "Funktionsstrukturen" et "Modulare Strukturen".

[6] Terminologie provenant de la norme X50-415 sur le "Management des systèmes – ingénierie intégrée – concepts généraux et introduction aux méthodes d'application", décembre 1994.

2. L'Ingénierie Système

L'ingénierie Système (IS) est une méthodologie interdisciplinaire récente, dont le but est de formaliser et de maîtriser les processus de conception, de réalisation et d'intégration de systèmes complexes (produits, systèmes de production...). Les principes de l'ingénierie simultanée peuvent être mis en œuvre dans le cadre de cette démarche. Différentes normes relatives à l'Ingénierie Système coexistent [EIA 632], [IEEE 1220], [ISO 15288]. Elles décrivent en termes de processus et d'activités génériques des principes d'organisation des projets et des pratiques jugées bonnes des métiers associés. Elles ont des champs d'application ou des approfondissements limités mais se complètent. Parmi les faits marquant l'histoire de l'IS, nous pouvons mentionner la constitution en 1990 de INCOSE (*International Council on Systems Engineering*) et en 1999, de l'AFIS (Association Française d'Ingénierie Système).

2.1. Principes

Selon l'AFIS, l'Ingénierie Système est une démarche méthodologique générale qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante aux besoins d'un client tout en satisfaisant l'ensemble des parties prenantes [AFIS]. Cette démarche permet d'intégrer les contributions de toutes les disciplines impliquées dans les phases de conception et d'intégration d'un système, en tenant compte des différentes exigences des parties prenantes (besoins, contraintes) intervenant au cours des différentes phases du cycle de vie d'un système (point de vue utilisateur). Grâce à la mise en œuvre systématique et coordonnée d'un ensemble de processus génériques par des équipes multidisciplinaires, l'IS permet la conception équilibrée d'une solution satisfaisant ces exigences, ainsi que des objectifs du projet en termes de coûts, délais, risques ... (point de vue du concepteur). Dans le cadre du développement d'un produit et de ses systèmes associés (système de conception, système de production, système de maintenance, …), les buts de cette démarche sont de décomposer un système complexe sur différentes strates, d'identifier et d'organiser les activités techniques, d'éviter les retours arrière tout en progressant avec assurance (gestion des risques et de la maturité de la conception sur chaque strate), de maîtriser les informations nécessaires à la réalisation et ainsi de réduire les délais et coûts de développement.

2.2. Modèles des processus d'ingénierie système

Dans cette partie, nous présentons deux modèles concernant les processus d'ingénierie système, ce qui nous permettra de réaliser un premier positionnement du processus de développement de l'architecture d'un système.

2.2.1. Modèle des processus génériques de l'IS

Parmi les processus génériques de développement (voir figure I-3), les normes d'IS distinguent les processus techniques, faisant passer des besoins à la solution, les processus de management et les processus facilitants. Au niveau de chaque strate de décomposition d'un système, les processus techniques sont mis en œuvre pour assurer progressivement l'exploration du problème (domaine fonctionnel) et la construction de la solution (domaine organique) [IEEE 1220]. Ces processus se décomposent en :

- processus d'ingénierie qui représentent la définition des exigences (analyse des besoins et spécification des exigences) et la conception de la solution (développement des architectures et spécification des sous-constituants), sur la branche descendante,
- processus de mise en œuvre qui traduisent la réalisation, l'intégration et la validation, sur la branche ascendante,
- processus supports de l'ingénierie qui sont transversaux à ces deux branches (analyse-système pour la recherche de compromis global et la justification des choix ; Vérification-Validation pour les activités de simulation, d'essais et de validations organique et fonctionnelle des solutions).

Au cours de la réalisation des activités de la branche descendante, l'acteur doit se projeter dans la réalisation des activités de la branche ascendante pour définir les plans d'intégration et de validation.

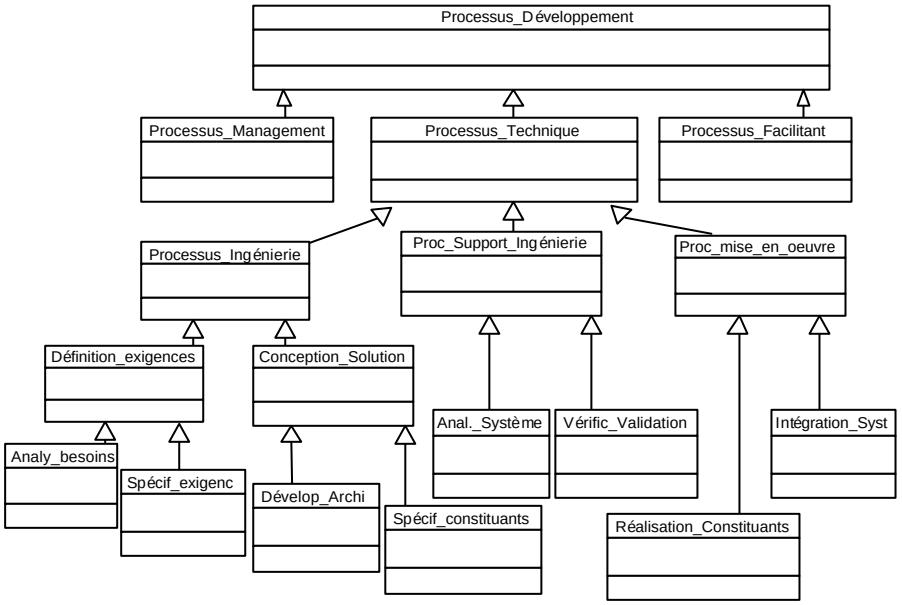


Figure I-3. Processus génériques en IS (tiré de [Bonjour et Dulmet, 2006])

2.2.2. Modèle du cycle de développement

La démarche de conception en Ingénierie Système peut être décrite par un cycle dit en V. Ce cycle de développement en V est composé principalement de trois phases (figure I-4):

- Une phase dite de conception et justification (branche descendante du V), faisant passer du besoin à une définition justifiée de la solution, de façon progressive et itérative, comportant :
  - Une analyse des besoins et contraintes des différentes parties prenantes (sur l'ensemble du cycle de vie) débouchant sur la spécification du problème et une spécification des exigences auxquelles doit satisfaire le système (définition des missions, fonctions de service, critères de performance, contraintes …),
  - une conception du système, consistant à élaborer :
    - l'architecture fonctionnelle qui est un arrangement de fonctions et leurs interactions, définissant le séquencement de leur exécution, les flux de données et de contrôle qui le conditionnent et les performances requises pour répondre aux exigences [IEEE 1220],
    - l'allocation (ou répartition, projection…) des fonctions sur les composants, préalablement identifiés,
    - l'architecture organique (technique) des constituants, supportant l'architecture fonctionnelle et définie par les exigences attribuées aux constituants et à leurs interfaces (qui supportent les interactions entre constituants). Le résultat est une spécification des exigences pour chaque constituant.
- Une phase d'acquisition ou de réalisation des constituants, sous-traitée aux métiers spécifiques concernés ou à des fournisseurs. Le développement des constituants complexes suit également un cycle de développement.
- Une phase dite d'intégration et de validation (branche ascendante), comportant :
  - La construction progressive du système par assemblage de ses constituants en validant que leurs interactions confèrent bien les comportements attendus aux différents niveaux d'assemblage.
  - L'intégration du système dans son environnement d'exploitation, suivie de sa qualification opérationnelle vérifiant qu'il répond au besoin opérationnel.

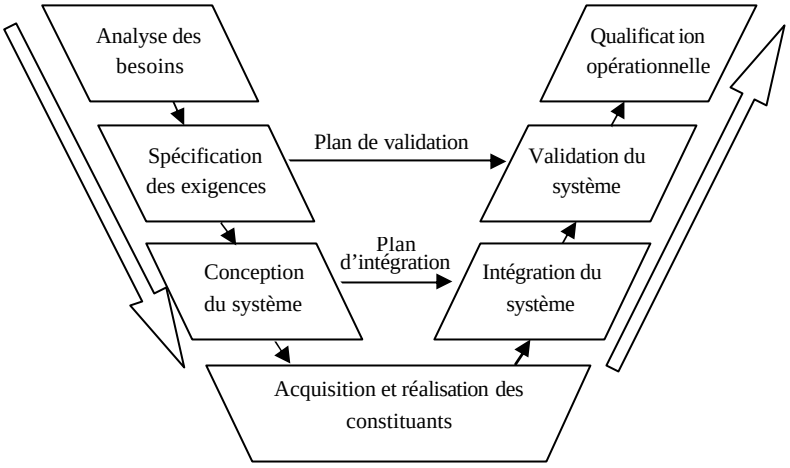


Figure I-4. Le Cycle en V de l'IS, issu de [Meinadier, 2002]

Compte tenu de la complexité des systèmes, la résolution du problème global nécessite sa décomposition en sous-problèmes (décomposition en blocs constitutifs) en tenant compte de leurs interactions, jusqu'à ce que ces sous-problèmes soient suffisamment simples pour leur trouver des solutions potentielles (composants existants ou à développer). On est donc conduit à itérer les activités de conception-justification puis d'intégration-validation à différents niveaux de granularité des constituants. Une définition appropriée de l'architecture prépare et conditionne le succès de l'intégration. La définition du système doit être justifiée par des activités d'analyse système et de vérification-validation.

L'Ingénierie Système (IS) fait référence à l'ensemble des deux branches du cycle de développement. Dans ce travail, nous nous intéressons principalement à la phase descendante du cycle en V et plus particulièrement à l'élaboration des architectures du système.

Notons enfin que la terminologie varie selon les auteurs ou les entreprises pour désigner la décomposition d'un système. Le terme « constituant » est le plus générique, il est parfois remplacé par sous-systèmes, organes, modules, blocs constitutifs, composants, pièces, etc. selon le niveau hiérarchique de décomposition considéré.

2.3. Le système et ses représentations

On distingue deux grands types de représentations d'un système [Meinadier, 2002] correspondant à une vue statique et à une vue dynamique. Dans ce travail de thèse, nous nous sommes surtout intéressés aux représentations statiques. Nous les développerons donc plus en détail dans les parties suivantes (2.3.1 à 2.3.4) et nous résumerons l'intérêt de la vue dynamique dans la partie 2.3.5. Les différents langages ou formalismes de modélisation, supportant ces différentes vues, seront présentées au chapitre II où nous justifierons notre choix concernant un outil de modélisation des architectures.

2.3.1. Vue externe (ou contextuelle) du système

En vue externe, le système doit remplir un ensemble de fonctions de service pour satisfaire les différentes parties intéressées par son utilisation. Elles traduisent l'action attendue ou réalisée par le produit pour répondre à un besoin d'un utilisateur donné (ou mission du système). Il faut souvent plusieurs fonctions de service pour répondre à un besoin. Elles sont quantifiées selon les performances attendues et spécifiées par leurs critères d'appréciation.

Il existe deux types de fonctions de service:

- les fonctions principales, correspondant au service rendu par le système pour répondre aux besoins,
- les fonctions contraintes, traduisant des réactions, des résistances ou des adaptations à des éléments du milieu extérieur.

A ce niveau, le système est représenté comme une boîte noire échangeant des flux avec son environnement, ainsi que par des scénarios d'échange (flux entrants et sortants) pour chaque fonction de service. Il est intéressant d'identifier des phases de vie du système ainsi que ses modes de fonctionnement pour chaque phase de vie.

2.3.2. Vue fonctionnelle interne

En vue interne, le système doit satisfaire un ensemble de fonctions techniques dont l'agencement restitue les fonctions de service. On parle ici d'arborescence puis d'architecture fonctionnelle du système, en cohérence avec Pahl et Beitz [1984] et Hubka et Eder [1988].

L'arborescence fonctionnelle est obtenue en décomposant itérativement les fonctions de service en sous-fonctions jusqu'à l'obtention de fonctions pour lesquelles on peut définir des solutions techniques [Meinadier, 2002]. Ainsi, une architecture fonctionnelle peut être obtenue à différents niveaux d'abstraction. A des niveaux de représentation très détaillés, la spécification des fonctions est de plus en plus liée à l'architecture physique du produit et renvoie de ce fait à une solution physique satisfaisant à la spécification de la fonction.

L'élaboration de l'architecture fonctionnelle est utile lorsqu'on cherche à optimiser l'organisation des sous-fonctions (par exemple, regroupement ou factorisation de module, dissociation pour faciliter les évolutions, redondance pour assurer la robustesse lors des évolutions ou pour la sûreté de fonctionnement) [Stone et al., 2000]. L'enchaînement des fonctions de l'architecture fonctionnelle permet de réaliser les processus de fonctionnement internes au système, lui conférant ainsi les comportements spécifiés par les fonctions de service et leurs scénarios. Cette architecture fonctionnelle est en théorie indépendante des choix de conception et est donc réutilisable pour des projets semblables. En pratique, la décomposition fonctionnelle ne se fait pas sans préjuger des choix technologiques et la

validation de l'architecture technique va nécessiter des itérations, pour modifier et/ou compléter les exigences du niveau et l'architecture fonctionnelle [Lartigue, 2003].

Les normes IEEE 1220 intègrent dans la définition de l'architecture fonctionnelle, des aspects dynamiques en tenant compte du séquencement de l'exécution des fonctions, des flux de données et de contrôle qui le conditionnent. Par la suite, nous parlerons d'architecture fonctionnelle pour désigner uniquement l'arrangement des fonctions et leurs interactions (tout en étant conscient que cela peut être considéré comme un abus de langage). Nous ajouterons l'adjectif « dynamique » pour préciser que nous intégrons les aspects dynamiques.

2.3.3. Vue organique (ou technique)

Dans la vue organique, le système est représenté comme un agencement de constituants (ou d'organes) réalisant par leurs interactions les fonctions identifiées dans l'architecture fonctionnelle. Cet agencement est appelé « architecture organique » où sont identifiés les constituants et les interfaces qui les relient (Figure I-5).

Certains auteurs comme Meinadier introduisent le concept d'architecture physique : « l'architecture technique devient l'architecture physique lorsque les organes sont définis », l'architecture organique étant alors considérée comme un agencement de constituants « théoriques ». Situant nos travaux dans les phases de conception préliminaire, nous ne ferons pas cette différence par la suite.

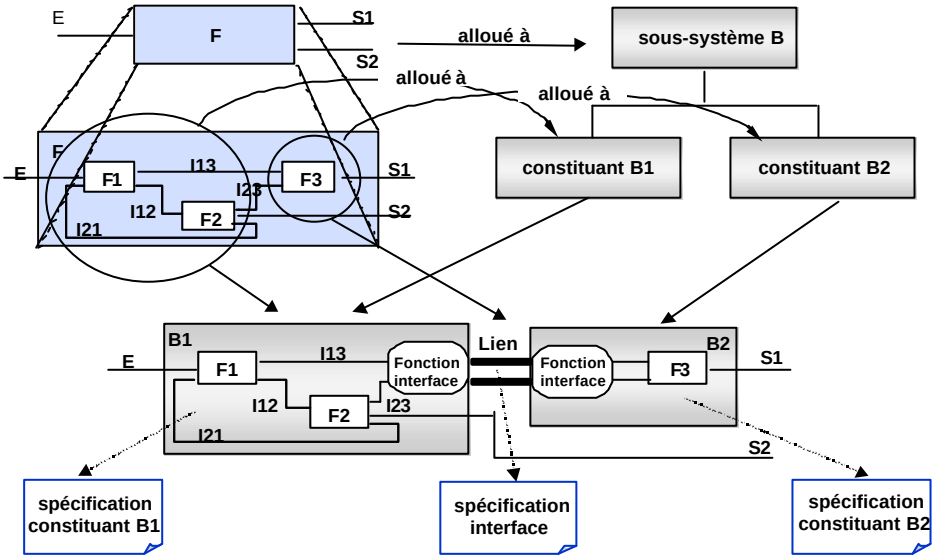


Figure I-5. Allocation des fonctions aux constituants organiques [Meinadier, 2002]

2.3.4. Vue dynamique d'un système

Pour structurer un système, il est possible de le décomposer en vue externe, en vue fonctionnelle interne et en vue organique, comme nous l'avons présenté précédemment. Ces vues sont des représentations statiques (ou structurelles) du système. Il est possible aussi de

recourir à des représentations dynamiques du système (vue dynamique) qui décrivent son évolution temporelle.

Le système peut être considéré comme un ensemble de processus de fonctionnement qui décrivent l'enchaînement des activités des constituants du système afin de restituer les fonctions de service et leurs performances. Ceci permet de représenter la dynamique de fonctionnement du système pour chaque mode de fonctionnement et pour chaque phase de vie du système. Cette approche généralisée de processus nous donne un point de vue nouveau sur le système où l'on peut suivre la transformation des entrées en sorties réalisant les fonctions de services.

En complément à la vue statique, la vue dynamique repose sur un enchaînement temporel de scénarios et processus, à différents niveaux d'invariance temporelle allant de l'évolution à long terme vers le fonctionnement : enchaînement des phases de vie, enchaînement des modes de fonctionnement dans chaque phase de vie, enchaînement des fonctions dans chaque mode de fonctionnement.

2.3.5. Synthèse sur le système produit

Dans ce travail, nous sommes intéressés prioritairement à la vue statique du produit. Dans le cadre de cette vue, nous avons introduit trois domaines : le domaine des exigences, le domaine des fonctions systèmes et le domaine des constituants. Dans la suite de ce mémoire, nous considérons que les exigences sont des contraintes qui s'appliquent au système produit et sont de ce fait externes au produit. Ainsi, comme nous l'avons représenté sur la figure I-6, le produit se décompose en vue fonctionnelle et en vue organique.

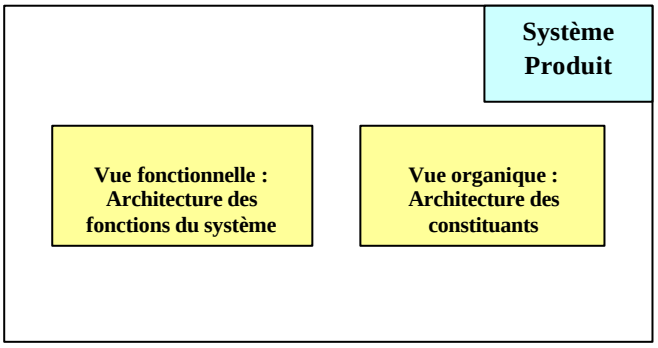


Figure I-6. Les vues fonctionnelle et organique du produit

3. Architecture du produit et conception modulaire

Dans les paragraphes précédents, nous avons souvent utilisé le terme d'architecture, tel que nous le rencontrons dans la littérature. L'utilisation de ce terme s'accompagne rarement d'une définition et d'une caractérisation rigoureuse. L'architecture étant le principal objet de notre travail, nous allons lui donner un cadre formel, utile pour notre modélisation.

Dans cette partie, nous revenons sur les définitions de l'architecture d'un produit. Nous précisons ensuite le processus de développement de l'architecture et une typologie des architectures. Nous présentons ensuite les travaux portant sur la conception modulaire et sur les métriques de la modularité.

3.1. Définitions de l'architecture d'un produit

Dans le cadre de l'ingénierie des produits et des systèmes complexes, nous avons trouvé plusieurs définitions de l'architecture d'un système.

Crawley et al. [2004] font référence à un système comme étant composé d'entités et donnent la définition suivante :

> *L'architecture d'un système est la description des entités qui le composent et des relations qui les lient.*

Maier et Rechtin [2000] quant à eux proposent une définition de l'architecture d'un système comme suit :

> *C'est la structure d'un produit, d'un processus ou d'un élément en terme de composants, connections et contraintes.*

Enfin, Meinadier [2002] propose la définition suivant :

> *Une architecture décrit une vision abstraite de la structure et du fonctionnement d'un système (produit ou processus). Elle est généralement constituée par un arrangement d'éléments (les éléments terminaux d'une arborescence de décomposition) à la fois sur le plan structurel (éléments et relations) et sur le plan temporel (dynamique de fonctionnement).*

Cependant, même si Meinadier sous-entend que la dimension temporelle fait partie intégrante de la définition de l'architecture, nous préférons adopter la définition plus générale proposée par Crawley et al. [2004]. Par ailleurs, il est à noter que l'architecture est une vision structurelle d'un système à un instant donné du projet et peut évoluer au cours du projet.

Les définitions du concept d'architecture du produit sont multiples mais elles se rejoignent toutes pour affirmer que l'architecture du produit se caractérise par : « la structure des composants et des interactions qui les lient » [Pimmler et Eppinger, 1994 ; Chen et Liu, 2005 ; Van Wie et al., 2001 ; Holtta, 2005].

Cependant, il existe d'autres définitions où l'accent est mis sur certains points caractéristiques du domaine de recherche. Ainsi dans le cadre de la conception concourante, nous retrouvons la définition suivante [Dahmus et al., 2000] : « l'architecture du produit est la structure des composants, leurs interactions et les principes qui gouvernent leur conception et évolution tout au long du cycle de vie. »

Au niveau international, les travaux de deux auteurs sont fréquemment cités par les chercheurs en architecture de produit : Ulrich et Eppinger. Ulrich [Ulrich et Tung, 1991] définit l'architecture du produit comme « une représentation générique des produits où un système conceptuel des composants physiques est associé à un système conceptuel des éléments fonctionnels afin de concevoir différents produits lors de l'étape de conception. »

Cette définition a été approfondie par [Ulrich, 1995], l'architecture du produit est alors :

- l'arrangement des fonctions ;
- l'allocation des fonctions aux composants ;
- la spécification des interfaces entre les composants.

En France, le point de vue de l'IS [Meinadier, 2002] sur l'architecture du produit rejoint celui d'Ulrich. On associe alors aux domaines fonctionnels et organiques la notion d'architecture, on utilise alors les terminologies d'**architecture logique ou fonctionnelle** et d'**architecture technique** ou organique. Au cours des décompositions conduisant à ces architectures, les exigences sont allouées (réparties) sur les éléments de la décomposition et les interfaces sont définies.

3.2. Processus de développement de l'architecture d'un système

En s'inspirant du schéma de la boucle descendante d'ingénierie décrite par l'IEEE 1220 (et reprise par Meinadier), nous proposons (figure I-7) un modèle du processus de développement de l'architecture d'un système. Ce processus est itératif et se répète à chaque niveau de décomposition du système. Les exigences de besoin de la strate supérieure (au plus haut niveau, les besoins du système) vont être tout d'abord analysées et détaillées grâce aux connaissances des concepteurs du niveau considéré. Les concepteurs procèdent alors à la définition de l'architecture fonctionnelle. La recherche de principes de solution et des choix possibles d'allocation permet progressivement de définir les constituants du niveau considéré. L'analyse des flux entre constituants permet d'aboutir à l'architecture technique. Les activités génériques d'analyse système (justification, évaluation de solutions alternatives et élaboration de compromis) et de vérification-validation ne sont pas représentées ici mais se déroulent en parallèle. Les choix de solutions techniques à un niveau peuvent contraindre les choix réalisés au niveau supérieur (par exemple, reconduction d'un constituant avec des interfaces figées). La spécification des constituants et de leurs interfaces permet d'allouer des exigences au niveau inférieur. Si le constituant est à son tour décomposable, le processus est itéré. Si le constituant est élémentaire ou est acheté, le processus de décomposition est terminé et est suivi par la réalisation ou l'acquisition des modules/composants. A chaque niveau de conception architecturale, des documents sont émis et seront utilisés pendant les activités cruciales d'intégration-validation (branche ascendante) correspondant à ce niveau.

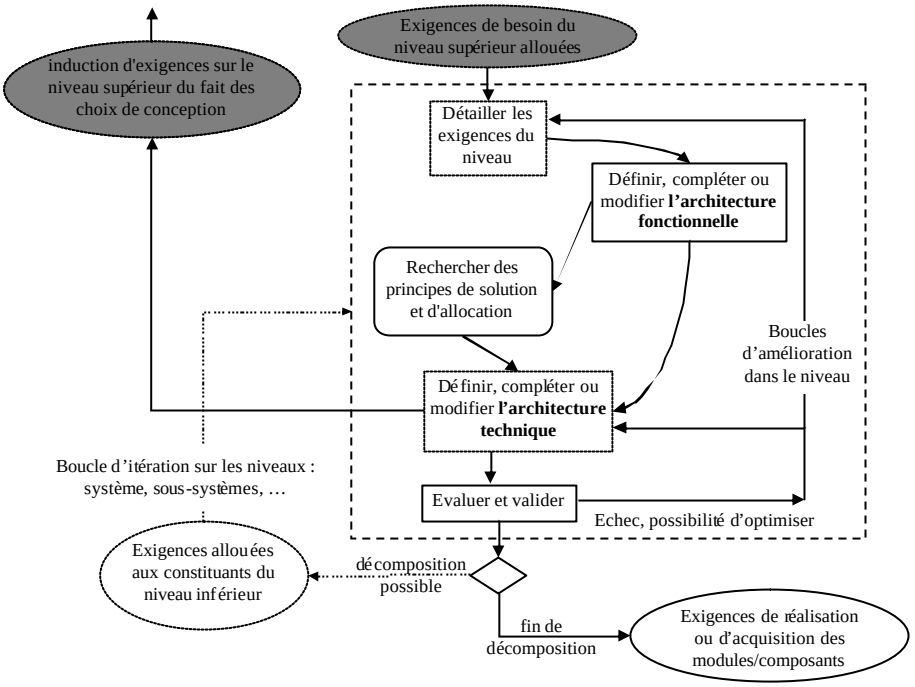


Figure I-7. Processus de développement de l'architecture d'un système (schéma adapté de l'IEEE 1220)

Pour appliquer ce processus de développement de l'architecture, nous avons besoin d'adopter une typologie qui permet de mieux caractériser les architectures fonctionnelles et organiques d'un système.

3.3. Typologie de l'architecture des produits

Ulrich [1995] distingue deux types d'architecture :

- L'architecture modulaire : une architecture est dite modulaire si on a un « *one to one mapping* » entre fonctions et modules (c'est-à-dire lorsqu'à une fonction ne correspond qu'un seul constituant physique) et si les interfaces sont découplées dans le sens où modifier un constituant ne demande pas de reconcevoir les autres interfaces**.** Il en résulte un découplage total entre les constituants.
- L'architecture intégrale (qu'on qualifiera aussi d'intégrative ou d'intégratrice) est une architecture où les constituants sont fortement couplés de telle sorte qu'un changement apporté sur un constituant nécessite l'apport d'adaptation sur d'autres constituants. Dans une architecture intégrale, les fonctions n'impactent pas directement un unique composant.

Cette typologie en deux points est prise comme référence dans des travaux ultérieurs sur la conception modulaire dont certains ont essayé de l'améliorer. Ainsi, Chen et Liu [2005] précisent que l'allocation « *one to one* » des fonctions vers les composants est dépendante du niveau de décomposition réalisé tant dans l'arborescence des fonctions que dans

l'arborescence du produit. Ils proposent alors de réaliser l'allocation d'un sous ensemble de fonctions vers un sous ensemble de composants.

Depuis la typologie d'Ulrich, il est communément admis qu'il y a un continuum entre une architecture entièrement modulaire et une autre, entièrement intégrale. Sans recourir aux méthodes de représentation des architectures qui seront traitées ultérieurement, il est facile d'imaginer qu'il existe une architecture hybride [Sosa et al., 2000] qui est composée à la fois de modules (groupements d'éléments) et d'éléments intégrateurs, un élément pouvant être une fonction, un constituant...

Ericsson et Erixon [1999] ajoutent à la définition d'Ulrich la condition qu'un module doit avoir peu d'interactions avec les autres modules ou le reste du produit. Ce dernier point est repris par Baldwin et Clark [2000] qui définissent un module comme « un ensemble d'éléments fortement liés entre eux et faiblement liés à d'autres éléments externes au module ». Par opposition aux modules, les éléments intégrateurs sont alors des éléments qui ne peuvent appartenir à aucun module, étant donné qu'ils interagissent fortement en nombre ou en intensité avec des éléments appartenant à plusieurs modules.

En étudiant la définition d'une architecture modulaire du produit, on remarque qu'elle a évolué durant cette dernière décennie. Avec Ulrich [1995], c'est l'allocation des fonctions aux constituants qui détermine la modularité de l'architecture et dans les travaux les plus récents [Baldwin et Clark, 2000 ; Eppinger et Salminen, 2001 ; Whitfield et al., 2002 ; Holtta et Salonen, 2003 ; Yang et al., 2004 ; Chen et Liu, 2005], c'est l'étude des interactions et des interfaces entre constituants qui déterminent la modularité d'une architecture.

La Figure I-8 montre l'exemple d'une architecture modulaire sur une perceuse où la finalité du travail était l'identification et la spécification des interfaces entre modules.

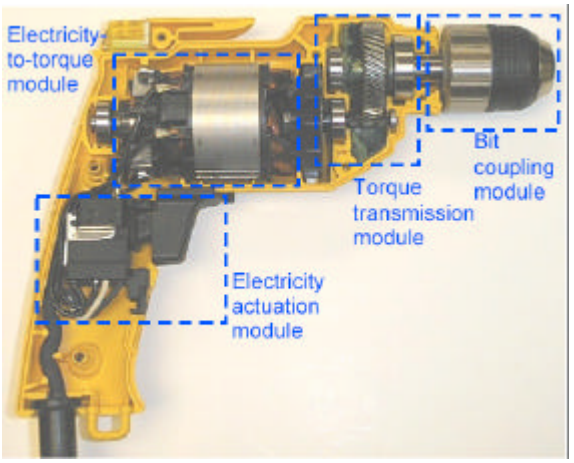


Figure I-8. Exemple d'architecture modulaire [VanWie et al., 2001]

Dans la continuité des travaux récents sur l'architecture du produit et dans le cadre de la typologie d'architecture d'Ulrich, nous considérons que l'architecture du produit est composée de :

- L'architecture fonctionnelle ;
- L'allocation des fonctions aux constituants ;
- L'architecture organique du produit.

La typologie d'Ulrich s'applique alors aux deux architectures du produit : à l'architecture fonctionnelle et à l'architecture organique. Cependant la caractérisation de ces architectures se fera essentiellement sur la base de l'étude des interactions internes à chaque domaine [Harmel et al., 2006b] en cohérence avec le positionnement de Baldwin et Clark [2000].

Sur la figure I-9, nous présentons notre vision de l'architecture du produit. Nous accordons une grande importance à l'utilisation de la double flèche pour l'allocation entre l'espace des fonctions et l'espace des constituants. En effet, nous considérons que du point de vue de l'architecture, les deux espaces se contraignent mutuellement sans hiérarchie, à moins que celle-ci ne soit voulue par le concepteur.

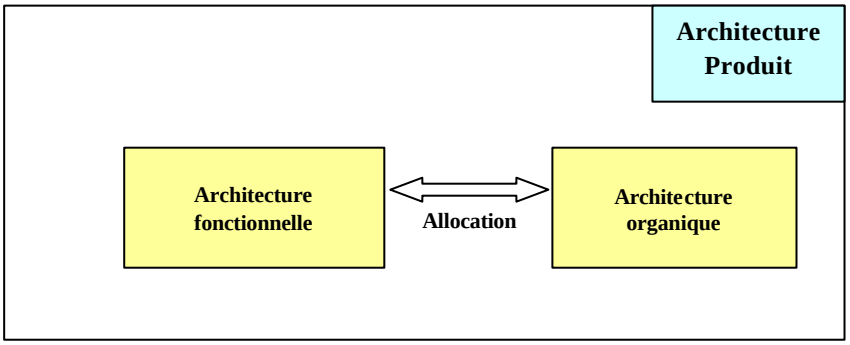


Figure I-9. Modélisation de l'architecture du Produit

3.4. Travaux existants sur l'architecture du produit

Le positionnement adopté dans le paragraphe précédent par rapport à la définition de l'architecture du produit et à la typologie adoptée fait consensus dans la plupart des travaux qui portent sur la modélisation de l'architecture du produit et de la conception modulaire.

Dans ce qui suit, nous donnons un aperçu sur le nombre et l'importance des travaux par sous-thèmes de recherche :

- L'élaboration de l'architecture du produit, relative à l'agencement hiérarchisé ou non d'éléments : [Dahmus et al., 2000 ; Whitfield et al., 2002 ; Yu et al., 2003 ; Sharman et Yassine, 2004 ;
- Hölttä, 2005; Jiao et al., 2006 ; Ulrich et Eppinger, 2000 ; Larses, 2005 ; Ulrich, 1995 ; Erixon, 1996 ; Balachandra, 2002 ; Holtta et Salonen, 2003 ; Pimmler, 1994 ; Sudjianto

et Otto, 2001 ; Huang, 2000 ; Stone et al, 2000]. Ces travaux nombreux traitent de la modélisation et génération de l'architecture du produit. Quelques uns [Oosterman, 2001] s'ouvrent à la modélisation du projet de conception pour lier l'architecture du produit aux autres caractéristiques du projet.

- La définition des interfaces et des flux d'échanges dans le produit [Blackenfelt et Sellgren, 2000 ; Van Wie et al., 2001 ; Chen et Liu, 2005 ; Andersson et Sellgren, 2004]. Ces travaux s'intéressent aux outils et méthodes de la conception détaillée du produit ainsi qu'aux échanges entre composants, permettant ainsi de faire le lien avec la vue dynamique du produit.
- La propagation d'une évolution de modules [Clarkson et al., 2004]. Très peu de travaux se sont intéressés à la robustesse d'une conception en modules et aux conséquences d'une évolution technologique, se traduisant par exemple, par le changement d'un module, sur l'architecture globale du produit.

L'abondance de travaux traitant de l'architecture du produit dénote de l'importance que tient cette problématique dans le domaine de l'ingénierie de la conception. Cependant, on peut nettement faire apparaître qu'il y a déséquilibre dans le traitement des sous-thèmes et que la modélisation de l'évolution des architectures est peu traitée.

Dans ce mémoire, nous nous intéressons à la fois à l'identification des architectures du produit et à la modélisation de leurs évolutions. Cependant, en considérant la modélisation de l'architecture du produit sous la forme d'architectures fonctionnelle et organique, nous nous démarquons vis-à-vis des travaux existants qui sont essentiellement axés sur l'architecture physique du produit.

Nous allons maintenant approfondir l'étude des architectures modulaires et de la modularité du produit.

3.5. Modularité et famille de produits

La modularité est une notion ambiguë qui a été employée de manière imprécise dans différents contextes. Dans son utilisation architecturale, ce terme est plus facile à définir à partir de son antonyme, c'est-à-dire modulaire est l'opposé d'intégré. Le mot « module » a comme définition[7] : « *assemblage compact fonctionnant comme composant d'une plus grande unité* ». L'adjectif « modulaire » est à la fois défini comme « *entier ou complet* ».

Une architecture entièrement modulaire est composée clairement d'un certain nombre de groupement de composants, ces groupements sont caractérisés par une forte densité des couplages entre les éléments qui les composent.

[7] Encyclopédie scientifique en ligne. http://www.techno-science.net

La notion de modularité est liée à la notion de diversification des produits et à la conception de familles de produits. En effet, des combinaisons et architectures appropriées de modules peuvent créer potentiellement un grand nombre de produits différents du point de vue du client, appartenant à ce qu'on appelle une famille de produits ayant certaines fonctionnalités communes et se différentiant à travers d'autres fonctionnalités [Langlois et Robertson, 1992 ; Sanderson et Uzumeri, 1990 ; Ward et al.,1995].

Il apparaît ainsi que la conception de produits modulaires est une démarche importante dans la stratégie de flexibilité [Sanchez, 1999] qui permet à une entreprise de s'adapter rapidement à l'évolution de la demande en mettant sur le marché de nouvelles variantes de ces produits en utilisant de nouvelles combinaisons des modules préexistants ou nouveaux.

La notion de modularité peut faire référence aux fonctions du produit. Pahl et Beitz [1996] proposent la classification suivante des modules :

- Un *Module Basique* est un module qui implémente une (ou plusieurs) fonction(s) fondamentale(s) du produit ou de la famille de produits.
- Un *Module Auxiliaire* est un module qui implémente une (ou plusieurs) fonction(s) auxiliaire(s) qui est utilisée en association avec les fonctions fondamentales pour créer de la variété dans la famille de produits.
- Un *Module Adaptatif* regroupe les fonctions qui permettent d'adapter un composant (ou un système) à un produit (ou un autre système). Les modules adaptatifs permettent de gérer les contraintes imprévisibles.
- Un *Non-Module* implémente des fonctions spécifiques dictées par les besoins des clients. Ces fonctions ne sont pas partagées avec d'autre produit de la famille.

Dans sa thèse, Mtopi [2006] utilise une autre typologie plus adaptée aux familles de produits. Il utilise les concepts suivants :

- Le « produit de base » rassemble tous les modules implémentant les fonctions principales de la famille et est considéré comme un « module composé ».
- un module indécomposable est dit « primitif »,
- un module commun, correspond à une ou plusieurs fonctions présentes dans tous les produits de la famille,
- un module distinctif introduit la variété dans la famille.

La modularité a aussi été associée à la notion de similitude [Huang et Kusiak, 1998] dans le sens où un module regroupe des éléments ayant des caractéristiques similaires (par exemple, des composants ayant la même géométrie, des fonctions liées à la même discipline scientifique comme la thermique, l'automatique).

3.6. Avantages et inconvénients de la conception modulaire

Plusieurs chercheurs [Huang, 2000 ; Oosterman, 2001 ; Balachandra, 2002 ; Holta et Salonen, 2003] ont fait l'inventaire des avantages, des inconvénients et des limites de la conception modulaire. Les points les plus importants sont synthétisés ci-dessous.

Les avantages les plus significatifs de la modularisation sont :

- Augmenter les variétés de produits en utilisant un nombre limité de composants [Ulrich 1995 ; Sanchez, 2002] : les produits peuvent être conçus de telle façon que la source de différenciation entre un ensemble de produits se concentre dans un composant modulaire.
- Permettre la personnalisation de masse : historiquement, les entreprises industrielles avaient soit des systèmes de production de masse ou de personnalisation. Avec la conception modulaire, il est possible d'utiliser des processus de production flexibles pour des volumes réduits avec une réduction réelle des coûts [Duray, 2000].
- Faciliter les stratégies de plateformes : la conception modulaire permet de concevoir une architecture de plateforme stable et d'introduire sur le marché des produits différents.
- Limiter les coûts de développement et d'industrialisation : une stratégie de conception modulaire permet de réduire les coûts en allouant certaines fonctions à des modules réutilisables par la suite.
- Permettre les économies d'échelle : c'est un principe simple d'économie qui stipule que lorsqu'on augmente les quantités de production, on réduit relativement les coûts. Or, la conception modulaire permet la réutilisation de certains composants et donc l'augmentation des quantités.
- Permettre une innovation technologique plus rapide : la modularisation permet à l'entreprise de se doter de processus de conception et d'industrialisation flexibles [Ericsson et Erixon, 1999]. C'est cette flexibilité qui permet à l'entreprise de répondre rapidement aux changements des marchés.
- Réduire les délais de développement (Time to market) : du moment que les interfaces entre les composants sont bien définies, il est possible de paralléliser les activités de conception. Ceci réduit les durées de développement et limite les phases de reconception.
- Faciliter la mise en place d'équipes de conception adaptée à l'architecture du produit. Les équipes seront de taille raisonnable et les échanges sont réduits vu que les interfaces sont bien spécifiées. [Sosa et al., 2000 ; Baldwin et Clark, 1997 ; Ulrich et Eppinger, 2000 ; Blackenfelt, 2001].

- Faciliter la maintenance, la réparation et le recyclage des produits tout au long du cycle de vie en permettant l'identification et l'accès à des modules [Newcomb et al., 1998 ; Dahmus et. al., 2000].

Mais la conception modulaire a aussi quelques inconvénients :

- Une utilisation intensive de la modularisation rend les produits tous semblables [Oosterman, 2001].
- La modularisation fait augmenter les risques de voir les produits copiés par les entreprises concurrentes [Yang et al., 2004].
- La conception initiale des modules est très difficile en raison de la complexité des contraintes sur les interfaces [Oosterman, 2001].
- Plusieurs auteurs affirment qu'une architecture modulaire est un frein pour la conception de produits performants à cause du compromis que réalise une architecture modulaire parmi tous les produits [Whitney, 2003].
- La conception modulaire peut engendrer des surcoûts, si par soucis de conserver les mêmes interfaces, on remplace un composant par un autre plus cher [Krishnan et Gupta, 2001].

Cette synthèse montre que les entreprises s'engageant dans une stratégie de conception modulaire de ses produits peuvent en tirer des bénéfices importants mais doivent être conscientes des limites et des risques de la modularisation.

Nous allons présenter maintenant les principales méthodes de modularisation, recensées dans les travaux de recherche en ingénierie de la conception.

3.7. Les méthodes de modularisation du produit

3.7.1. Introduction

Les outils d'aide à l'identification des modules dans un produit sont peu nombreux dans la littérature de l'ingénierie de la conception. Parmi elles, trois méthodes sont fréquemment référencées :

- Méthode heuristique pour l'architecture des fonctions,
- Méthode MFD (Modular Function Deployment),
- Méthode de "clustering" d'une matrice structurelle de conception (en anglais DSM pour Design Structure Matrix).

Une étude comparative de ces trois méthodes a été réalisée par Holtta et Salonen [2003]. Pour notre part, dans cette partie, nous ne présenterons que les deux premières méthodes. La

troisième étant celle que nous avons utilisée dans ce travail, elle sera traitée en détail dans le chapitre III.

Il est à noter que d'autres techniques, telles que la méthode de Hatley/Pirbhai [Zakarian et Rushton, 2001] et les graphes d'interaction [Kusiak et Huang, 1996], ont été également employées pour la conception de produits modulaires. Les travaux de Messac et al. [2002] ont utilisé des techniques de programmation pour formuler une fonction de pénalité pour une famille de produits. Siddique et Rosen [1999] ont proposé une approche de grammaire de graphes pour identifier la ressemblance dans une famille des produits et d'autres ont également utilisé des Algorithmes Génétiques (AG) [D'Souza et Simpson, 2003] pour rechercher les modules communs d'une plateforme.

Il est important de préciser que toutes ces méthodes sont susceptibles d'avoir comme résultats des architectures modulaires non-optimales et non-uniques. De plus, certaines méthodes sont liées à la définition de l'architecture du produit comme étant l'allocation des fonctions aux composants.

3.7.2. Méthode des heuristiques pour l'architecture des fonctions

Avant de présenter cette méthode, nous allons définir succinctement ce qu'est une heuristique.

Une heuristique est une démarche (ou un algorithme) de résolution de problème, connue pour fournir un résultat proche de l'optimum (une « bonne solution »). La décision de proposer à l'utilisateur un choix parmi plusieurs bonnes solutions à la place d'un optimum illusoire est à conseiller, surtout lorsque les critères à optimiser sont multiples et mal explicités. Les heuristiques formalisent souvent l'utilisation de règles empiriques, tirées de l'expérience, du bon sens ou d'analogies. Comparées à un algorithme optimal, elles présentent l'avantage d'être plus facilement exploitables et compréhensibles par les utilisateurs qui peuvent alors mieux s'approprier les résultats proposés. Les heuristiques trouvent leur place dans les problèmes complexes qui nécessitent l'exploration d'un grand nombre de solutions possibles, car elles permettent d'atteindre rapidement des solutions qui ont le plus de chances de donner une réponse satisfaisante.

La méthode des heuristiques pour générer l'architecture des fonctions a été développée initialement par Stone et al., [1998, 2000]. Elle est basée sur trois heuristiques séparées qui permettent d'identifier les modules d'un produit. Son point de départ est la décomposition fonctionnelle développée par Pahl et Beitz [1996]. Elle est née d'observations d'un engin de maintenance. Un ensemble de fonctions de ce produit interagit par des flux d'information pour former des modules physiques. Ces observations se résument comme suit :

1- un flux peut traverser un produit en restant inchangé : ce sont les flux dominants.

2- un flux peut se ramifier pour former des chaînes de fonctions indépendantes.

3- un flux peut être converti en un autre.

Ces observations ont conduit Stone et ses coauteurs [1998, 2000] à la formulation de trois heuristiques :

- Heuristique des flux dominants : elle examine les flux à travers une arborescence fonctionnelle, elle suit les flux jusqu'à ce qu'ils sortent du système ou subissent une transformation. Elle conclut à : s'il existe un ensemble de fonctions à travers lesquelles un flux passe jusqu'à la sortie ou la transformation, alors ces fonctions forment un module.
- Heuristique des flux ramifiés : elle examine les flux divergents ou convergents issus de chaînes de fonctions parallèles. Chacune de ces chaînes peut devenir un module dont l'interface correspond au point de branchement de la chaîne.
- Heuristique du module de conversion-transmission : elle examine les flux convertis. Ces flux sont convertis au niveau de certaines fonctions. On considère alors ces fonctions comme étant des modules à elles seules. Si dans une même chaîne de fonctions il y a plusieurs fonctions de conversion ou de transmission, alors ces fonctions délimitent un module de conversion-transmission.

Zamirowski et Otto [1999] proposent trois heuristiques additionnelles pour identifier les fonctions partagées dans un produit unique ou dans une famille de produit et les fonctions uniques propres à un produit donné. Ainsi, les fonctions qui partagent des flux similaires ou qui apparaissent plusieurs fois peuvent être regroupées pour former un unique module. Les fonctions uniques sont celles spécifiques à un seul produit, elles peuvent être regroupées pour former un module.

Nous remarquons que les principaux critères de modularisation sont fonctionnels et indépendants de l'architecture physique du produit. Cette méthode est présentée comme étant idéale pour la conception de famille de produits [Holtta et Salonen, 2003] mais elle présente le défaut d'être dépendante de l'interprétation et de la décision humaine. Cette méthode ne peut pas être automatisée et de ce fait ne peut être appliquée sur des produits complexes [Otto et Wood, 2001]. Pour des présentations plus approfondies de cette méthode et surtout de la démarche de mise en œuvre, nous conseillons la lecture des travaux de Dahmus et al. [2000], Otto et Wood [2001] et Holtta et Salonen [2003].

3.7.3. Méthode MFD

La méthode MFD (Modular Function Deployment) est une méthode de modularisation plus orientée management qu'ingénierie. Elle a été développée par Erixon [Erixon, 1996 ; Ericsson et Erixon, 1999]. Cette méthode est aussi basée sur la décomposition fonctionnelle et elle est dédiée à la modularisation d'un produit unique.

La méthode MFD est basée initialement sur 12 « pilotes de modularité », ce nombre monte jusqu'à 23 avec Blackenfelt [2001]. Le tableau I-1 présente et explique les 12 pilotes initiaux.

Développement du produit	Reconduction	Une partie du produit peut être réutilisée et reconduite dans plusieurs produits.
	Evolution technologique	Les pièces sont susceptibles de subir des changements en raison des demandes de clients ou des innovations technologiques. Il est important de pouvoir adapter les interfaces de sorte que la nouvelle technologie puisse être introduite sans remettre en cause l'ensemble du produit.
	Modifications planifiées	L'entreprise a l'intention de développer et de changer certaines parties du produit.
Variabilité	Spécifications variables	Pour prendre en compte la personnalisation des produits efficacement, un concepteur devrait tâcher d'assigner toutes les variations à un petit nombre de composants.
	Style	Les modules de style contiennent les composants qui peuvent être altérées pour créer des variations du produit.
Production	Unité en commun	Les composants et fonctions en commun entre plusieurs produits permettent le partage d'une unité de production.
	Processus et/ ou organisation	Des pièces exigeant le même processus de production sont groupées ensemble.
Qualité	Essais séparés	La possibilité de tester séparément chaque module avant la livraison à l'assemblage final peut contribuer à l'amélioration de la qualité.
Achats	Disponibilité des fournisseurs	Modules standard d'achat disponibles chez les fournisseurs externes.
Après-vente	Service et maintenance	Les composants nécessitant une maintenance peuvent être groupés ensemble pour former un module qui facilite les opérations de remplacement et de réparation.
	Amélioration	Il faut donner aux clients la possibilité de faire évoluer leur produit.
	Recyclage	Le nombre de matériaux dans chaque module devrait être limité. Les matériaux facilement recyclables peuvent former un module séparé.

Tableau I-1. Les pilotes de la modularité dans la méthode MFD

Dans sa stratégie de développement de produits modulaires, une entreprise peut utiliser un ou plusieurs des pilotes présentés dans le tableau I-1. Elle applique ensuite la démarche de MFD qui comprend cinq étapes :

1- Clarifier la spécification du produit en utilisant une matrice QFD [Akao, 1990] avec les pilotes de modularité qui remplacent les besoins des clients.

2- Analyser les fonctions et sélectionner les solutions techniques.

3- Identifier les modules possibles en utilisant la matrice MIM (Module Indication Matrix)

4- Evaluer les concepts en testant les interfaces entre les modules.

5- Améliorer chaque module.

L'identification des modules de la méthode MFD se fait en classant les fonctions selon un score croissant. Les fonctions participant à un même pilote de modularité sont candidates pour

former un module. Blackenfelt [2000] propose un rapprochement entre la méthode MFD et l'outil DSM, que nous décrirons ultérieurement.

La méthode MFD présente l'avantage de laisser le choix aux ingénieurs de définir les stratégies de modularisation en jouant sur les pilotes. Ce même avantage est l'un des inconvénients de la méthode qui reste dépendante des ingénieurs dans le choix des modules et de leurs tailles, ceci a pour conséquence de limiter la possibilité d'automatiser la méthode et d'avoir la même architecture d'un utilisateur à un autre (faible reproductibilité).

3.7.4. Besoin d'une autre méthode de modularisation

Les deux méthodes de modularisation présentées ci-dessus montrent certes plusieurs avantages mais aussi certaines limites :

- Elles dépendent des choix des ingénieurs à certains moments de la démarche de modularisation.
- Elles ne peuvent pas être automatisées et sont difficilement exploitables sur des exemples complexes où une assistance par un traitement informatique serait souhaitable.
- Elles sont essentiellement adaptées à l'espace fonctionnel.

Ces limites font que, comme d'autres chercheurs, nous opterons pour une autre méthode de modularisation, à savoir la méthode de *clustering* des matrices DSM. Celle-ci permet à la fois de modéliser les architectures par les DSM et de générer des architectures modulaires « satisfaisantes », à travers l'utilisation d'un algorithme de *clustering*.

3.8. Les métriques de la modularité

Quand on parle de modularité, on est en droit de se poser la question : à quel point un produit est-il modulaire ? Afin de quantifier la modularité, plusieurs mesures et métriques ont été développées. Rappelons cependant que plusieurs études ont conclu à l'absence de pertinence de la comparaison des modularités entre des produits différents [Gershenson et al., 2004]. Les mesures développées servent alors à comparer différentes architectures d'un même produit.

Les mesures que nous présentons dans ce paragraphe sont classées selon une complexité croissante.

Blackenfelt [2000] utilise deux métriques pour mesurer la modularité d'un produit : le MI (Module Independence) [Newcomb et al., 1998] et l'ARP (Average Ratio of Potential) dont les expressions sont :

$$MI = \sum_{m=1}^{n} \frac{in_m}{tot} \qquad \textbf{Eq.I-1}$$

$$ARP = \sum_{m=1}^{n} \frac{1}{n} \frac{in_m}{in_{pot}} \qquad \textbf{Eq.I-2}$$

Avec :

n :	Le nombre de modules
m :	L'indice dun module
in_m :	La somme des interactions à l'intérieur du module m
in_{pot} :	La somme des interactions à l'intérieur d'un module pondérée par un facteur à fixer

La différence principale entre ces deux métriques est que l'ARP est maximal pour une architecture composée de plusieurs petits modules.

Les métriques utilisées par Blackenfelt ne prennent pas en compte les interactions externes aux modules, de ce fait la pertinence des métriques est très limitée.

Une autre mesure de la modularité est le SMI [Holtta et al., 2005] pour « Singular Value Modularity Index », la mise en place de cet indicateur nécessite une étape intermédiaire qui est la diagonalisation de la DSM du produit. On a alors :

$$DSM = U\Sigma_{DSM}V^T \qquad \textbf{Eq.I-3}$$

Avec

$$\Sigma_{DSM} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & s_N \end{bmatrix} \qquad \textbf{Eq.I-4}$$

Où les s_i sont les valeurs propres classés dans un ordre décroissant.

On définit alors SMI par :

$$SMI(\Sigma_{DSM}) = 1 - \frac{1}{Ns_1}\sum_{i=1}^{N-1} s_i(s_i - s_{i+1}) \qquad \textbf{Eq.I-5}$$

La valeur de SMI est encadrée par 0 et 1, un SMI proche de 1 indique un fort degré de modularité, un SMI proche de 0 définit une architecture intégrale.

Se basant sur les valeurs propres de la DSM, cette métrique ne permet pas de faire le lien entre la taille des DSM, la taille des modules, leurs densités et la densité des interactions externes. L'interprétation de cette métrique nécessite de se référer à la DSM initiale.

Guo et Gershenson [2004] ont étudié huit métriques de modularisation et proposent une mesure de la modularité basée sur les intensités des interactions entre éléments dans un système. Cette métrique (Eq. I-6) se compose de deux parties : une partie positive qui donne un poids aux interactions internes aux modules et une partie négative qui pénalise les interactions externes aux modules.

$$\frac{\sum_{k=1}^{M_m} \frac{\sum_{i=n_k}^{m_k}\sum_{j=n_k}^{m_k} R_{ij}}{(m_k - n_k + 1)^2} - \sum_{k=1}^{M_m} \frac{\sum_{i=n_k}^{m_k}\left(\sum_{j=1}^{n_k-1} R_{ij} + \sum_{j=m_k+1}^{N_c} R_{ij}\right)}{(m_k - n_k + 1)(N_c - m_k + n_k - 1)}}{M_m} \qquad \textbf{Eq.I-6}$$

Avec :

n_k :	index du premier élément du k^{ème} module	N_c :	nombre total de composants dans le produit
m_k :	index du dernier élément du k^{ème} module	R_{ij} :	la valeur de l'interaction entre les éléments i et j
M_m :	nombre total des modules dans le produit		

Cette mesure peut être utilisée avec toutes les DSM composants, elle présente l'avantage de prendre en compte les interactions extérieures aux modules. Cependant, Wietfield et al. [2002] utilisent une métrique proche de celle de Guo avec des pondérations plus pertinentes (le nombre d'interactions disponibles). Cette mesure de la modularité est l'indicateur MSI pour Module Strength Indicator. Le MSI s'applique à un module et se compose de deux parties (Eq.I-7 et Eq. I-8)

$$MSI_i = \frac{\sum_{i=n_1}^{n_2}\sum_{j=n_1}^{n_2} DSM(i,j)}{(n_2-n_1)^2-(n_2-n_1)} \qquad \textbf{Eq.I-7}$$

$$MSI_e = \frac{\sum_{i=0}^{n_2}\sum_{j=n_1}^{n_2} DSM(i,j)+DSM(j,i)}{2\times(n_1\times(n_2-n_1))} + \frac{\sum_{i=n_2}^{N}\sum_{j=n_1}^{n_2} DSM(i,j)+DSM(j,i)}{2\times((N-n_2)\times(n_2-n_1))} \qquad \textbf{Eq.I-8}$$

$$MSI = MSI_i - MSI_e \qquad \textbf{Eq.I-9}$$

Avec :

$DSM(i,j)$	Valeur de l'interaction entre l'élément i et j dans la DSM
n_1	Index du premier élément du module
n_2	Index du dernier élément du module
N	Le nombre total délément

L'indicateur MSI permet alors de mesurer le degré de modularité de chaque module en comparant les interactions à l'intérieur et à l'extérieur du module, le degré de modularité augmente lorsque le nombre d'interactions internes au module croît et que le nombre d'interactions externes décroît. Cet indicateur permet ainsi à l'architecte système d'évaluer la modularité de chaque module et de simuler plusieurs configurations. Ce dernier indicateur semble le plus complet pour évaluer la modularité d'une architecture. Une dernière remarque concernant cette métrique : les auteurs proposent les indices n_1 et n_2 et affirment que le nombre d'éléments est n_2-n_1, pour que cette affirmation soit vraie il faut que n_1 soit l'indice du dernier élément dans le module précédent.

Cependant, cet indicateur comme les précédents ne peut pas être utilisé pour optimiser d'une manière automatique une architecture car ils tendent tous à créer un module unique représentant tout le système.

Quelques travaux de recherche se sont intéressés à la comparaison de certaines métriques présentées dans ce paragraphe [Holtta et al., 2005 ; Larses et Blackenfelt, 2003]

4. De l'architecture du produit vers l'architecture de l'organisation du projet

Dans cette partie, nous allons étendre les concepts définis pour l'architecture de l'organisation du projet.

4.1. Ingénierie Système appliqué au projet

Les normes qualité ISO9000-version 2000 définissent un projet comme un « processus unique (particulier) qui consiste en un ensemble d'activités coordonnées et maîtrisées comportant des dates de début et de fin. Ce processus est entrepris dans le but d'atteindre un objectif conforme à des exigences spécifiques telles que les contraintes de délai, de coûts et de ressources. » Nous adaptons cette définition en considérant qu'un projet est l'instanciation et la réalisation d'un processus unique.

Dans cette partie, nous allons montrer que l'Ingénierie Système peut s'appliquer sur le système-projet. Nous allons définir et expliquer ce qu'est l'architecture de l'organisation du projet.

4.1.1. Les trois domaines du projet

Nightingale [2000] et Eppinger et Salminen [2001] ont mis en évidence qu'un projet de conception de produit repose sur 3 domaines inter-reliés : le produit, le processus de conception et les acteurs de conception (équipe ou concepteur individuel). Un acteur a la responsabilité de réaliser une (ou plusieurs) tâche(s) de conception, qui contribue(nt) à la définition justifiée du produit. Dans ce travail, nous nous rapprochons de cette vision du projet. Cependant, nous introduisons le domaine de l'organisation du projet. Le paragraphe qui suit explique notre positionnement.

4.1.2. Différentes vues de l'organisation du projet

Dans le paragraphe relatif à l'IS, nous avons intentionnellement conféré un caractère générique au système. La première instanciation d'un système concerne le produit. Une seconde instanciation concerne le système-projet. L'organisation du projet peut ainsi être considérée sous des vues différentes, en cohérence avec la définition de Meinadier [Meinadier, 2002, p53] que nous reprenons ci-dessous :

- « une vision contextuelle de l'entreprise échangeant des flux avec les organismes de l'environnement selon des scénarios d'échange (par exemple, flux de matières, produits ou services et flux monétaires compensatoires),

- une décomposition structurelle en fonctions et sous-fonctions (l'organigramme fonctionnel),
- une décomposition en niveaux d'invariance temporelle,
- des processus enchaînant des activités exercées par les fonctions, décrivant le fonctionnement de l'organisation,
- une architecture physique formée des entités de l'organisation et de leurs interfaces,
- une subdivision en un système opérant et un système de pilotage généralement hiérarchisé en fonction des niveaux temporels ».

Les tâches de développement peuvent être déterminées à partir d'une combinaison entre les activités génériques de développement et les constituants de l'« arborescence technique du produit »[8] à réaliser (organigramme du produit, ou *PBS, Product Breakdown Structure*). Le résultat de cette combinaison est représenté sous forme d'une arborescence ou d'une succession de tâches et est appelé « organigramme des tâches » (ou *WBS, Work Breakdown Structure*)[9] [PMI, 2001 ; PMI, 2004]. Cette arborescence est ensuite utilisée pour décrire le projet comme un réseau de processus enchaînant des tâches. C'est l'architecture fonctionnelle du système-projet. Les processus (de fonctionnement du système-projet) doivent être coordonnés en fonction des dépendances entre tâches, de leurs conditions de fin et en vue de respecter les objectifs de réalisation. L'architecture organique consiste à allouer des tâches et des ressources à des entités organisationnelles, ou équipes, (analogie avec la projection des fonctions sur les constituants du produit) qui auront alors la responsabilité de l'accomplissement de ces tâches. Nous utiliserons le terme « acteur » pour désigner soit une entité organisationnelle ou une équipe (acteur collectif) soit un concepteur (acteur individuel).

Notons que l'architecture fonctionnelle doit avoir une certaine robustesse, en cas d'évolution d'une tâche ou d'un acteur au cours du projet, ainsi que d'un projet à un autre (à un certain niveau d'abstraction). Par ailleurs, l'allocation des tâches à des acteurs peut être contrainte par des aspects non techniques mais économiques, politiques ou juridiques, comme par exemple, l'existence de Métiers ou d'entités juridiques, des décisions d'*outsourcing*.

Si l'ingénierie de la conception fait partie de la productique, les travaux de recherche traitant de l'organisation d'un projet concernent aussi d'autres domaines scientifiques, telles que les sciences de gestion, les sciences cognitives, la sociologie ou la théorie des organisations [Simon, 1997] [Midler, 1998] [Hatchuel et Weil, 2002] [Lefebvre et al., 2002] [Perrin, 2001].

[8] Terminologie provenant de la norme X50-400 sur le "Management des systèmes – Référentiel cadre – Lignes directrices pour l'utilisation des méthodologies du management de projet", décembre 1994.

[9] Definition of Work Breakdown Structure according to [PMI 2001]. A deliverable-oriented grouping of projects elements that organizes and defines the total scope of the project. Each descending level represents an increasingly detailed definition of the project work.

Notre travail, reste dans le cadre de la productique, nous l'axons sur la vue structurelle de l'organisation de projet.

Le terme « organisation » a de multiples significations : une action d'organiser un système, un résultat de l'organisation, une entité « entière » comme une entreprise ou une association, … Nous choisissons la définition suivante de l'organisation du projet (ou structure organisationnelle, pour certains auteurs) : c'est le résultat de la structuration du projet qui permet de définir les processus, les acteurs et leurs responsabilités (missions, liens hiérarchiques), les moyens et procédures de communication.... Nous parlerons de l'architecture de l'organisation du projet pour désigner les architectures fonctionnelle et organique, présentées ci-dessus, tout comme nous parlons d'architecture du produit.

4.2. Vers l'architecture de l'organisation du projet

Il est possible d'étendre la définition de la modularité basée sur l'étude des interactions entre éléments [Ericsson et Erixon, 1999 ; Baldwin et Clark, 2000 ; Sosa et al., 2000] à tout système et ainsi à l'organisation du projet.

Dans le paragraphe précédent, nous avons introduit différentes vues de l'organisation du projet et nous avons étendu les concepts d'architectures fonctionnelle et organique des produits à l'organisation du projet de conception. Ainsi, nous considérons les architectures du domaine fonctionnel de l'organisation, à savoir les processus et du domaine organique, à savoir les acteurs du projet. Nous définissons l'architecture de l'organisation du projet comme composée d'une architecture fonctionnelle et d'une architecture organique (Figure I-10).

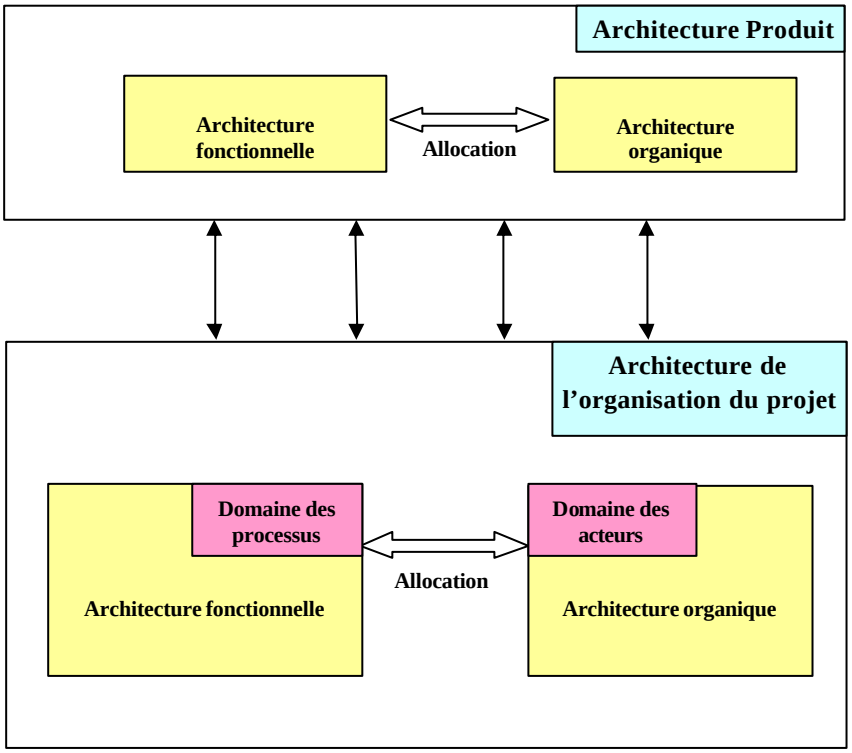


Figure I-10. De l'architecture du produit à l'architecture de l'organisation du projet

Comme nous avons choisi de caractériser les architectures des systèmes que nous étudions en fonction des interactions internes à ces domaines, nous pouvons étendre les notions d'architectures modulaires et d'éléments intégrateurs à toutes les architectures que nous traitons, y compris celles de l'organisation du projet.

Les méthodes de modularisation que nous avons introduites précédemment sont liées à l'architecture du produit. Nous présenterons dans le chapitre suivant un outil matriciel de représentation des interactions applicables au système projet de conception. Cette méthode de représentation nous donnera la possibilité d'utiliser un algorithme de clustering comme méthode de modularisation applicable à tout système (*Cf* Chapitre III).

4.3. Pilotage du processus de définition de l'architecture du produit

Dans le paragraphe 3.2, nous avons présenté un processus de définition de l'architecture du produit. Ce processus inclut des tâches d'analyse des exigences, d'amélioration dans le niveau ou entre niveaux (boucles, itérations), ainsi qu'une tâche d'évaluation et de validation. Par ces tâches, il est ainsi couplé à un processus de pilotage opérationnel. Cependant, pour que ce processus existe, il a fallu le concevoir, c'est-à-dire définir ses objectifs ou missions et le décomposer en un ensemble de sous-processus. Les normes de l'Ingénierie Système en fournissant un ensemble de processus génériques aident à la bonne structuration et à l'agencement des processus de conception.

4.4. Les niveaux de pilotage du projet d'IS

Meinadier [2002] soutient que : « la planification d'un projet consiste à rendre cohérentes entre elles trois logiques, dans le respect des objectifs et contraintes du projet :

- la logique du produit à réaliser (arborescence des produits),
- la logique temporelle des processus (enchaînement des activités),
- la logique de la structure organisationnelle du projet. »

Et par extension, la conduite de projet a pour objectif de veiller à la réalisation et au maintien de cette cohérence tout au long du projet face aux aléas.

Nous souhaitons affiner cette vision en distinguant trois niveaux importants dans le pilotage d'un projet [Bonjour et Dulmet, 2006] :

- Pilotage stratégique : le projet est décomposé en phases qui constituent son cycle de vie. Le passage d'une phase à la suivante donne lieu à une revue de projet planifiée où des décisions critiques sont prises : décisions d'engagement, de poursuite avec maintien des objectifs ou avec révision des moyens, d'itération pour amélioration de la solution ou

pour révision des objectifs, ou décision d'arrêt. Durant ces revues, le chef de projet rend compte au comité de pilotage de l'avancement du projet en termes d'objectifs et de moyens mis en œuvre.

- Le pilotage organisationnel (ou définition de l'architecture du système-projet) : elle consiste à analyser les besoins et à spécifier les objectifs et missions du système-projet (études d'opportunité), puis à définir le processus de conception comme un enchaînement de tâches à réaliser (architecture fonctionnelle du projet) pour atteindre ces objectifs et enfin à affecter un acteur à la réalisation de chaque tâche afin d'obtenir un enchaînement des flux d'informations entre acteurs (architecture organique du projet). Les termes pour désigner ces activités varient fortement selon les auteurs. Nous parlerons de pilotage organisationnel du projet. Quand le projet est lancé, ces activités de structuration peuvent être itérées périodiquement pour adapter l'organisation du projet, en fonction des écarts observés ou de nouvelles connaissances acquises en cours de projet. En effet, on ne peut espérer une organisation détaillée des tâches et des ressources, définie une fois pour toute et stable tout au long du projet. Le projet n'a pas un caractère déterministe mais fortement incertain (problème incomplètement défini au départ, constante évolution de l'environnement …). Les architectes systèmes et chefs de projets n'acquièrent donc que progressivement les connaissances sur le projet qui leur permettent de l'organiser au mieux.

- Le pilotage opérationnel (ou conduite de projet) : elle consiste à gérer les activités dans le plan du projet, à les ordonnancer, à maîtriser au mieux les dérives et modifications dues aux aléas, de manière à respecter les objectifs du projet et à garantir la cohérence de la maturité à l'intérieur d'une strate ainsi que la progression entre les strates. En effet, la mise en œuvre de rangs d'exigences entraîne qu'il n'est pas nécessaire d'attendre la fin de la conception dans une strate pour enclencher la conception dans la strate aval. La gestion de l'information est ainsi une activité importante pour le bon déroulement du projet. Les activités de pilotage opérationnel permettent d'intégrer progressivement les différentes contributions des acteurs du projet et de fournir les informations nécessaires à la validation de ces résultats, lors des revues de projet.

Certains auteurs adoptent une structuration semblable du système de pilotage [Génelot, 1992], en particulier pour la conception et le pilotage de systèmes de conception [Girard, 2004].

De nombreux travaux portent sur :

- le pilotage opérationnel des projets de conception : par exemple, ordonnancement de projet [Giard, 1991], [Herroelen et Leus, 2005] ou aide à la décision multi-critère pour l'évaluation des résultats d'un projet [Yannou et Bonjour, 2006] ;

- le pilotage stratégique et les décisions d'outsourcing : par exemple, choix de projets dans un portefeuille de projets [Levine, 2005].

Notre travail s'intègre dans le cadre du pilotage organisationnel, c'est-à-dire de la conception des architectures du produit et de l'organisation du projet.

4.5. Travaux sur les architectures de l'organisation du projet

Les travaux de recherche qui se sont intéressés à l'architecture de l'organisation du projet, telle que nous l'avons définie précédemment, peuvent être classés en trois grandes thématiques :

- Travaux traitant uniquement de la décomposition de l'organisation en équipes de conception [Morelli et al., 1995 ; David et al., 2002 ; Tseng et al., 2004 ; Fitzpatrick et Askin, 2005] ;
- Travaux traitant de l'optimisation des processus par la réorganisation des équipes de conception : [McCord et Eppinger, 1993 ; Eppinger et al., 1994; Chen and Lin, 2003 ; Braha, 2002] ;
- Travaux traitant de la correspondance entre les architectures du produit et de l'organisation : [Gulati et Eppinger, 1996 ; Lockledge et Salustri, 2001 ; Sosa et al., 2000 ; 2003 ; 2004 ; Sanchez et Mahomey, 1996; Browning, 2001; Oosterman, 2001].

Ces trois familles de travaux ont des objectifs très différents. De ce fait, les outils et les approches de l'architecture de l'organisation sont différents. Ainsi, les travaux appartenant à la première classe empruntent certains concepts aux sciences humaines, en étudiant les techniques de communication et les modes de travail. Ils ont pour but de définir un découpage adéquat (voire optimal) des équipes de conception. Quant aux travaux traitant de l'optimisation des processus de conception, ils incorporent un aspect temporel sous forme de contraintes de précédence et proposent de réorganiser les équipes de conception pour limiter les retours-arrières des flux d'information ou faciliter les chevauchements entre tâches. Ces deux premiers types de thématiques sortent du cadre de notre travail.

Nous positionnons une partie de nos contributions sur la troisième thématique qui s'intéresse à la correspondance entre l'architecture du produit et celle de l'organisation. Cependant, au lieu de représenter cette correspondance comme étant une propagation d'un domaine vers un autre, nous proposons de modéliser dans le chapitre V cette correspondance comme étant une dépendance mutuelle qui nécessite la coévolution des architectures du produit et de l'organisation du projet.

5. Synthèse sur la conception conjointe des architectures du produit et de l'organisation du projet

Dans ce chapitre, nous nous sommes positionnés comme suit :

- Le produit comme l'organisation du projet sont des systèmes qui peuvent être décomposés d'une manière itérative en sous-systèmes inter-reliés.
- L'architecture du produit est définie par la caractérisation des architectures fonctionnelle et organique ainsi que par leur couplage (ou allocation).
- L'architecture du produit est modélisée selon la typologie d'Ulrich [1995]. Par extension, cette typologie nous permet de caractériser l'architecture du produit sous la forme de modules et d'éléments intégrateurs :
    - Baldwin et Clark [2000] définissent un module comme étant « un ensemble d'éléments fortement liés entre eux et faiblement liés à d'autres éléments externes au module ».
    - Un élément intégrateur est un élément qui interagit fortement en nombre ou en intensité avec des éléments appartenant à plusieurs modules. Par conséquent, il est préférable qu'il n'appartienne à aucun module car il crée la cohésion du système dans sa globalité.
- L'architecture de l'organisation du projet est définie par la caractérisation de l'architecture des processus et l'architecture des acteurs du projet (structuration en équipes).
- La typologie des architectures et notre définition des modules et des éléments intégrateurs sont étendues à l'architecture de l'organisation du projet.
- Dans le cadre du pilotage organisationnel, nous nous intéressons à la conception des architectures du produit et de l'organisation ainsi qu'à leurs évolutions au cours du projet.

Précisons avant de conclure que récemment d'autres travaux au niveau national ont fait le parallèle entre la conduite du projet et la conception du produit [Baron, 2005] et [Gutiérez-Estrada, 2007] et ont proposé une autre approche formelle des couplages.

Avant de présenter l'outil de modularisation que nous avons développé dans ce travail et son application sur l'identification des architectures (chapitre IV), nous avons besoin d'adopter un modèle de représentation des architectures qui soit compatible avec notre problématique. Dans le chapitre qui suit, nous présentons les outils matriciels que nous avons utilisés pour modéliser l'architecture du produit et de l'organisation du projet.

CHAPITRE II

CHAPITRE II

MODELISATION MATRICIELLE DE L'ARCHITECTURE D'UN SYSTEME

De tout temps, les chercheurs ont travaillé sur les méthodes et outils pour mettre en œuvre les concepts qu'ils proposent. Nous venons de clarifier dans le chapitre précédent notre positionnement vis-à-vis du concept d'architecture et de l'Ingénierie Système.

Dans ce chapitre, nous allons introduire un premier outil qui nous permettra de faire l'articulation entre notre définition de l'architecture et les méthodes de modularisation que nous traiterons dans le chapitre III.

Avant de présenter l'outil sur lequel notre choix s'est porté, nous allons présenter succinctement d'autres méthodes de représentation des architectures du produit.

1. Les modèles de représentation des architectures

Les représentations référencées dans ce travail s'adressent à deux types d'architecture : l'architecture fonctionnelle et l'architecture physique. Il y a de multiples modèles et méthodes pour représenter l'architecture d'un produit. Aussi nous en présenterons succinctement quelques-uns, qui sont :

- soit à la base des autres modélisations (SA ou SADT, diagrammes d'états-transition),
- soit les plus utilisés en Ingénierie Système (arborescence hiérarchique, FFBD, SA-RT).

Mentionnons d'abord l'existence de méthodes globales de modélisation des systèmes informatiques ou technologiques, comme :

- la méthode SAGACE, développée au CEA [Penalva, 1997], qui adopte une approche systémique globale pour la représentation des systèmes complexes sous de multiples points de vue. En effet, elle couple deux axes d'analyse du système à trois dimensions (d'abord fonctionnel, structurel et décisionnel, et ensuite, action, fonctionnement et évolution), ce qui donne 9 représentations possibles d'un système et une grande richesse de modélisation. Même si leur sémantique est plus riche, les diagrammes clés dans Sagace (diagramme de transaction de flux ; diagramme d'interaction ou de transition entre activités) sont très proches des diagrammes SADT.
- le langage SysML, construit à partir du langage UML2, évolution d'UML. SysML est amené à devenir une référence internationale en modélisation pour l'IS [OMG, 2006] SysML est composé de différents types de diagrammes représentant différentes vues du système : diagramme des exigences (nouveau type), diagrammes comportementaux, dont les cas d'utilisation et les diagrammes de séquences (représentant des scénario d'échanges entre le système et les entités de son environnement), et les diagrammes structurels dont les diagrammes de définition des blocs et les diagrammes paramétriques (nouveau type) pour faire le lien entre la vue statique et la vue dynamique du système.

Des représentations matricielles, comme les DSM (Design Structure Matrix), peuvent aussi être utilisées. Nous les détaillerons dans la partie 2.

1.1. Modélisation par une arborescence hiérarchique

Le moyen le plus simple pour représenter une architecture est probablement l'arborescence hiérarchique. Dans un arbre, un système est décomposé en sous-systèmes qui seront à leur tour décomposés jusqu'à l'obtention du niveau de décomposition voulu. Cette représentation est possible pour décomposer les fonctions ou décomposer les constituants du système.

En enrichissant sa sémantique, une arborescence hiérarchique peut permettre de représenter la diversité dans une famille de produits. La Structure Générique d'un Produit (SGP) d'une

famille de produits fait référence à l'organisation générique de tous les modules qui peuvent se trouver dans la famille [Jiao et al., 2000]. En se rapprochant du concept de nomenclature générique, un module générique est un module abstrait qui représente une famille de modules concrets similaires. Nous distinguons en différentes catégories de modules concrets (figure II-1):

- des modules communs qui appartiennent à toutes les variantes de la famille de produits,
- des modules distinctifs qui rendent les variantes du produit différentes les unes des autres,
- des modules composés (qui peuvent être décomposés !) et par opposition, des modules primitifs.

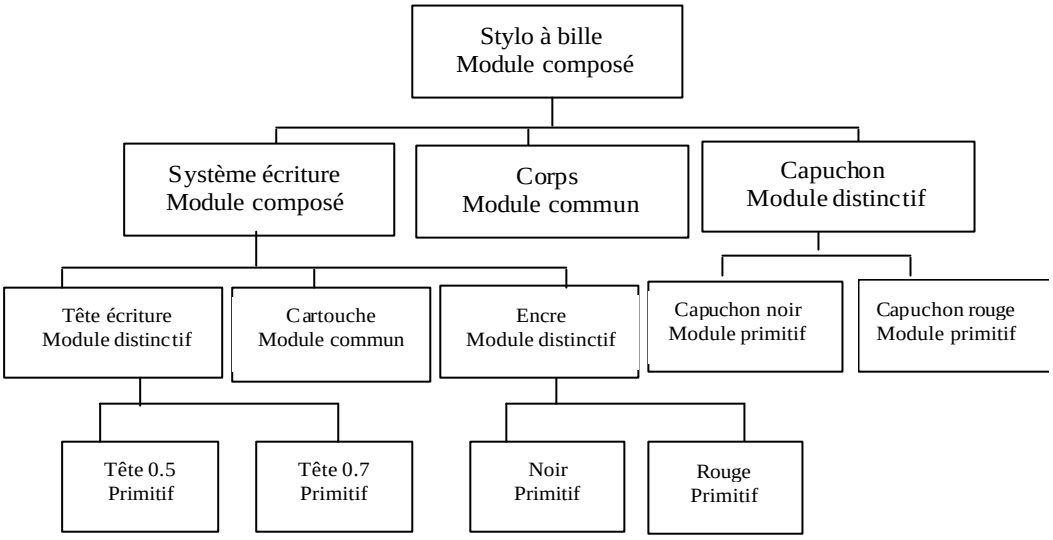


Figure II-1. Arborescence d'une structure générique de produit [Mtopi, 2006]

1.2. Modélisation de l'architecture fonctionnelle en vue statique

Les formalismes permettant de représenter l'architecture fonctionnelle de l'organisation d'un projet (c'est-à-dire un processus vu comme un enchaînement de tâches) peuvent être utilisés pour représenter l'architecture fonctionnelle du produit.

Différents types de modèles sont fondés sur des diagrammes représentant les fonctions et sous-fonctions transformant des flux entrants en flux sortants. Ces diagrammes sont obtenus par décomposition successive du système à différents niveaux hiérarchiques. Le système est d'abord vu comme une boîte noire échangeant des flux avec son environnement. Ensuite les fonctions du système sont décomposées en sous-fonctions transformatrices des flux de données ou d'objets (matière, énergie, donnée).

Dans les diagrammes de flux (ou flots) de données (DFD, Data Flow Diagramm), un élément de modélisation supplémentaire (double traits) permet d'indiquer un stockage de données. La

simplicité de ces diagrammes les rend facilement exploitables pour représenter des systèmes traitant de flux de données.

Dans la méthode SADT, il s'agit de représenter un enchaînement d'activités (boîtes) transformant des flux entrants en flux sortants (flèches horizontales), au moyen de ressources ou organes (flèche verticale montante) et sous l'effet de contrôle (flèches verticales descendantes) qui peuvent être considérées comme des déclencheurs ou des paramètres de régulation (figure II-2). Ce dernier élément de modélisation correspond à un premier pas vers les aspects dynamiques. SADT est très utilisé dans la modélisation des processus. Parfois utilisée sans le concept de ressources, elle permet aussi de représenter simplement une architecture fonctionnelle d'un produit.

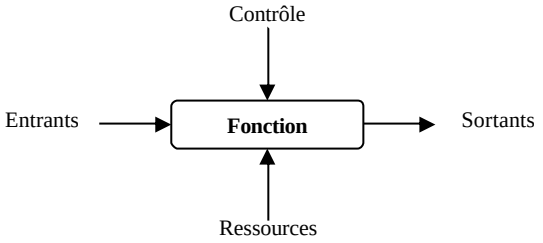


Figure II-2. Modèle de fonction par SADT

Dans la figure II-3, nous avons repris deux représentations possibles de l'architecture fonctionnelle d'un lave-linge. La représentation du dessus expose la décomposition sous forme d'arborescence hiérarchique des fonctions du lave-linge. La représentation du dessous représente par un diagramme de flux simplifié l'enchaînement des tâches sur le cycle de fonctionnement. Ces deux représentations nous renseignent à deux niveaux différents sur l'architecture fonctionnelle du lave-linge.

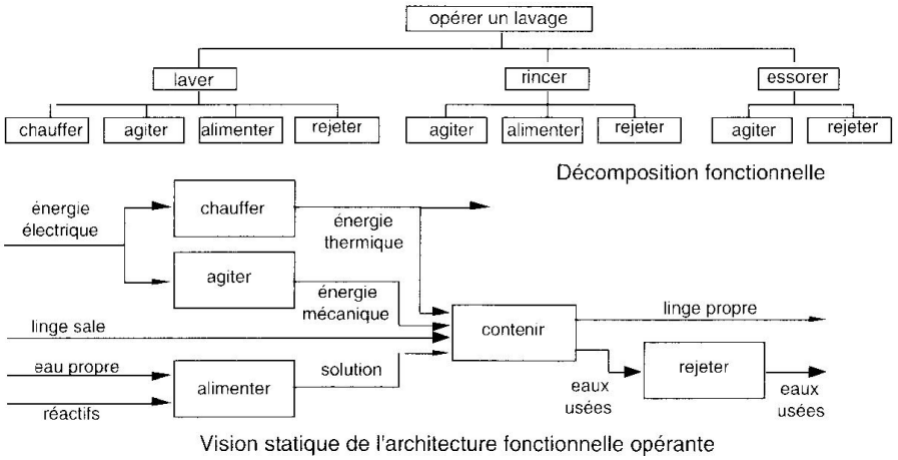


Figure II-3. Deux visions de l'architecture fonctionnelle d'un lave-linge [Meinadier, 1998]

Dans la figure II-4, nous avons repris de Stone et al. [2000] la modélisation par SADT de l'architecture fonctionnelle d'un tournevis électrique. Cette modélisation a été enrichie d'une

part en représentant différent types de flux liant les fonctions et d'autre part en représentant directement sur les diagrammes les modules fonctionnels qu'ils ont identifiés. Nous verrons ultérieurement qu'avec l'outil de modélisation que nous allons adopter la représentation des modules va être simplifiée et surtout formalisée.

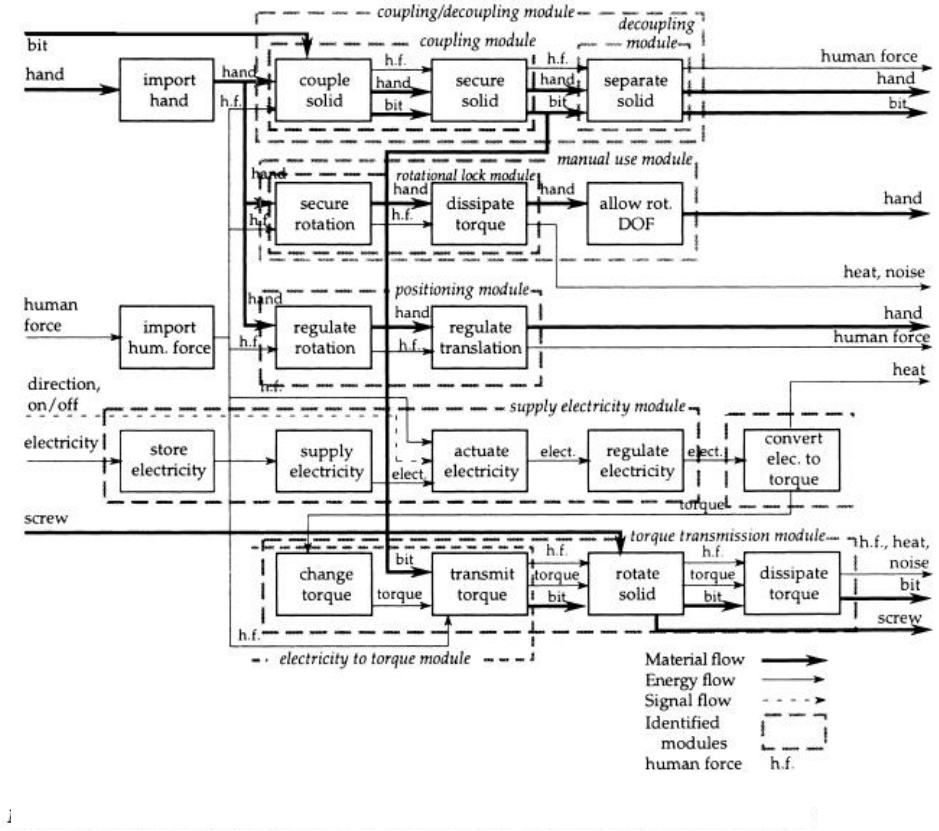


Figure II-4. Utilisation de SADT pour modéliser le fonctionnement d'un tournevis

1.3. Modélisation de l'architecture fonctionnelle dynamique

Différents formalismes existent pour intégrer une vue fonctionnelle et une vue dynamique du système (on parle souvent de modèles comportementaux). Les plus utilisés en Ingénierie Système [Meinadier, 2002] sont :

- les diagrammes de scénarios d'enchaînement de fonctions (ou diagrammes de flux fonctionnels, FFBD),
- les diagrammes d'états-transitions (statecharts) [Ward et Mellor, 1985], qui ont eu diverses extensions [Harel, 1987].

1.3.1. La méthode FFBD

Un diagramme de flux fonctionnels (FFBD, *Functional Flow Bloc Diagram)* peut représenter une architecture fonctionnelle (ou un scénario d'enchaînement fonctionnel). L'architecture fonctionnelle (ou logique) peut prendre en compte les flux de matière, d'énergie et

d'information [Pahl et Beitz, 1996]. Les FFBD sont utilisés pour décrire le contenu et l'ordre de déroulement des fonctions. Les fonctions sont représentées par des blocs, qui se succèdent les uns aux autres dans l'ordre indiqué par les flèches. Lorsque deux opérations sont réalisées en parallèle, elles sont reliées par deux ET (ou deux AND). Il peut également y avoir plusieurs alternatives, que l'on représente alors à l'aide de OU (ou de OR). Deux méthodes permettent de représenter la répétition : soit à l'aide d'une boucle d'itération (IT), qui peut préciser le nombre de répétition, soit à l'aide d'une boucle conditionnelle (LP comme Loop), qui se terminera sur une condition de fin de boucle (LE comme Loop End).

Par rapport aux FFBD simples, les EFFBD (Extended FFBD) introduisent une vision dynamique supplémentaire en indiquant les échanges de flux entre les fonctions. Les flux d'entrée peuvent être utiles ou nécessaires à la réalisation de la fonction. Dans certains cas, ils peuvent également être des « déclencheurs » (trigger) de la fonction.

Cette représentation peut être aussi utilisée pour représenter la décomposition du produit en composants.

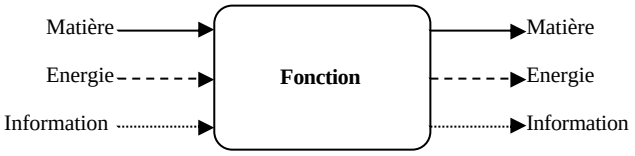


Figure II-5. Représentation d'une fonction par FFDB

1.3.2. Les diagrammes d'états et la méthode SA-RT

Les diagrammes état-transition modélisent les enchaînements d'états auxquels sont attachées les activités fonctionnelles. Ils mettent en relief les événements déclenchant l'enchaînement des fonctions. Ils permettent de représenter la dynamique d'évolution dans les systèmes temps réel. Ils modélisent les synchronisations série et parallèle des fonctions. On associe aux états les activités opérantes des processus de fonctionnement et aux transitions, dont les conditions de franchissement sont exprimées par des équations logiques, les actions de commande associées au passage dans le nouvel état et donc au déclenchement des activités correspondantes. Les diagrammes d'états (statecharts) de Harel présentent une amélioration dans la précision des diagrammes état / transition classiques.

SA-RT (Structured Analysis Real Time, [Hatley et Pirbhai, 1987]) est une méthode particulièrement bien adaptée à la spécification fonctionnelle dans les phases amont. Elle représente une extension temps réel de la méthode DFD. Ses auteurs ont ajouté à cette description des diagrammes de flots de contrôle et des spécifications de contrôle permettant de représenter les informations qui activent ou désactivent les processus représentés dans les DFD. Ward et Mellor [1985] préconisent l'utilisation de diagrammes états-transitions pour mettre en relief les événements déclenchant les processus.

Baron [2005] fait une étude comparée de SA-RT et d'UML pour la modélisation des systèmes mécatroniques. Elle note que SA-RT est adapté à la vue dynamique des systèmes embarqués mais détaille peu la conception de l'architecture matérielle des systèmes.

La figure II-6 représente le scénario de démarrage d'un moteur thermique par un digramme d'état.

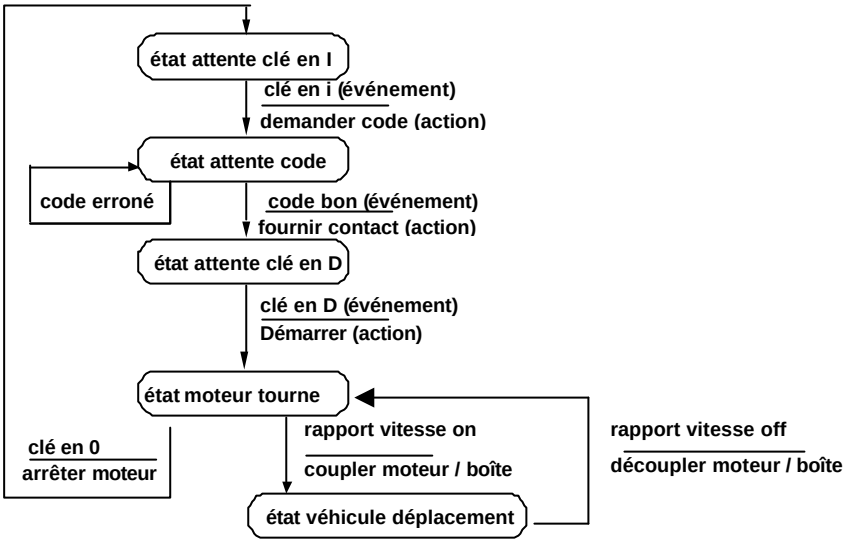


Figure II-6. Exemple simple de scénario de démarrage moteur

1.4. Modélisation de l'architecture organique

Nous n'avons pas identifié d'outils dédiés spécifiquement à la modélisation de l'architecture organique (mis à part des représentations géométriques ou cinématiques par des outils de CAO). Cependant, les outils de décomposition hiérarchique et les diagrammes de flux de données sont très fréquemment utilisés pour représenter l'architecture organique d'un produit.

Sur la figure II-3, nous avons représenté l'architecture fonctionnelle d'un lave-linge. Sur la figure II-7, nous représentons son architecture organique en utilisant le même type de représentation.

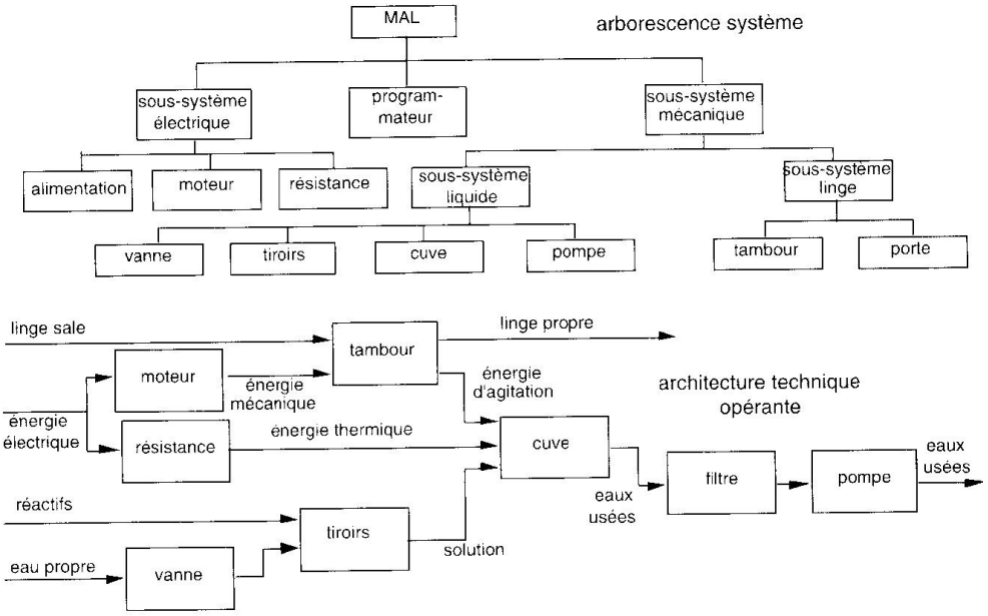


Figure II-7. Deux visions de l'architecture organique d'un lave-linge [Meinadier, 1998]

1.5. Synthèse sur la modélisation des architectures

Les méthodes présentées sont des méthodes plus proches de la représentation fonctionnelle du produit que de sa représentation physique avec prise en compte des interfaces. Elles ne sont pas automatisées pour permettre la modélisation et la simulation des architectures de systèmes complexes.

L'étude bibliographique développée succinctement ici montre que pour représenter des modules, il est nécessaire d'avoir un outil de modélisation des architectures qui soit à la fois visuel, simple d'utilisation, qui se prête à la programmation et à la simulation. Ces caractéristiques nous ont confortés dans le choix de l'outil DSM basé sur une représentation matricielle des architectures. La partie qui suit est dédiée à la présentation de l'outil DSM en mettant en avant sa position parmi les outils matriciels de modélisation des architectures.

2. Modélisation des architectures par des outils matriciels

2.1. Introduction

Les outils matriciels ont pour principal objectif la facilitation de l'analyse des interactions des systèmes complexes dans le cadre des activités suivantes [Malmqvist, 2002] :

- Modélisation des produits [Erixon, 1998],
- Analyses des interactions entre les constituants d'un produit [Pimmler and Eppinger, 1994],
- Modélisation de la conception [Suh, 1990],

- Analyse des impacts des évolutions [Clarkson et al., 2004].

Un outil matriciel de modélisation représente une vue d'un système sous une forme matricielle. Cette représentation peut servir à l'analyse du projet à différents niveaux : fonctionnel, interfaces, etc… Cette analyse se fait par le biais de plusieurs types d'algorithmes tels que « le clustering », le partitionnement, etc…

2.2. Classification des méthodes matricielles

Malmqvist [2002] a classé les méthodes matricielles en trois classes :

- Matrices pour le niveau élément : ces matrices représentent des relations entre des éléments/des composants d'un projet. On identifie deux sous catégories à savoir : les matrices inter-domaines et les matrices intra-domaines.
- Matrices pour le niveau système : ces matrices permettent de lier des caractéristiques ou des propriétés du système, il n'y a pas la notion de décomposition qui existe dans les matrices éléments où tous les éléments forment le produit ou le projet. Ces matrices système servent par exemple pour les plates-formes de produit [Ulrich and Eppinger, 2000].
- Méthodologies matricielles : ces méthodologies utilisent un ensemble de méthodes matricielles avec une cohérence interne qui leur permet de traiter certains problèmes complexes de conception.

Nous allons approfondir dans ce qui suit la présentation des matrices pour le niveau élément. Comme nous l'avons présenté plus haut, ces matrices peuvent être de deux types :

- Les matrices intra-domaines : elles représentent des couplages entre éléments de même type, entre composants par exemple. Ces représentations matricielles peuvent être utilisées à des différents niveaux d'abstraction : systèmes, composants, attributs. Les matrices intra-domaines peuvent être utilisées pour étudier les interactions entre des propriétés/attributs, entre fonctions, entre sous-systèmes/composants/paramètres, entre acteurs et bien sûr entre processus. Les éléments sur la diagonale de ces matrices sont sans signification. Ces matrices sont en effet les DSM citées précédemment, de ce fait nous utiliserons dans la suite la dénomination de **DSM** pour qualifier les matrices intra-domaines (aussi appelées matrices de couplage par Meinadier).
- Les matrices inter-domaines : elles représentent des couplages entre éléments de types différents et de ce fait appartiennent à différents domaines. Les matrices de la conception axiomatique [Suh, 1990] sont un exemple de matrices inter-domaines, ces matrices permettent de visualiser les relations entre des fonctions et des paramètres de conception. Un autre exemple est celui de la méthode QFD (Quality Function Deployment) [Akao, 1990], ces matrices représentent les liens entre un composant et les

processus de fabrication. Dans la suite de ce travail, nous appellerons ce type de matrice, des **Matrices d'Incidence (MI)** (aussi appelées matrices de traçabilité ou de projection, par Meinadier ou matrices d'incidence et d'allocation en mathématiques).

2.3. Représentation par les matrices binaires

L'utilisation des matrices dans le domaine de modélisation des systèmes remonte aux travaux de Warfield dans les années 70 [Warfield, 1973] et de Steward dans les années 80 [Steward, 1981]. Cependant, ce n'est que dans les années 90 que cette méthode a reçu une attention de plus en plus grande et une diffusion importante et ce, sous l'impulsion principale des chercheurs du MIT *(Massachussets Institute of Technology)* qui ont utilisé cette méthode dans le domaine de la modélisation des architectures en conception.

A l'origine des matrices binaires utilisées par Warfield et Steward, on trouve la théorie des graphes. Berge [1958] définit un graphe comme étant un ensemble de sommets, un ensemble d'arêtes et une fonction d'incidence qui associe deux sommets à chaque arête. Un graphe est dit orienté si on différencie le sommet de départ et le sommet d'arrivée. Une représentation des graphes orientés est donnée dans la figure II-8.

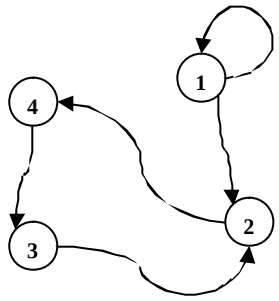


Figure II-8. Exemple de graphe

Un graphe peut être aussi représenté par une matrice d'adjacence. On définit alors la matrice d'adjacence par : un élément de la matrice vaut 1 si l'arc existe et 0 sinon. Ainsi le graphe précédent peut être représenté par la matrice de la figure II-9. Les sommets de départ sont en ligne et les sommets d'arrivée en colonne.

	1	**2**	**3**	**4**
1	1	1	0	0
2	0	0	0	1
3	0	1	0	0
4	0	0	1	0

Figure II-9. Traduction en matrice d'adjacence

Dans le cas d'un graphe non orienté, une liaison entre deux nœuds i et j implique que les éléments de la matrices d'adjacence d'indices ij et ji sont non nuls et identiques. La matrice est symétrique.

Les matrices binaires de Warfield et Steward sont des matrices d'adjacence sauf qu'on n'associe aucune signification aux éléments en diagonale (les boucles refléxives sont interdites), c'est pourquoi ils sont soit laissés vides, soit noircis (éventuellement, on rappelle le nom de l'élément sur la diagonale pour faciliter la lecture d'une matrice de grande taille).

Dans la littérature, les matrices binaires sont représentées généralement par une des formes (équivalentes) illustrées sur la figure II-10.

	A	B	C
A		X	
B			X
C	X		

	A	B	C
A		1	0
B	0		1
C	1	0	

Figure II-10. Deux formes de représentations binaires

Nous distinguons deux principales classes de matrices binaires :

- La première classe correspond aux matrices symétriques, c'est-à-dire à des graphes non orientés (tableau II-1). Ces matrices, les plus souvent utilisées, permettent de représenter des interactions symétriques sans relation de hiérarchie ou de précédence entre les constituants du système. Dans cette classe, deux blocs basiques permettent de modéliser les interactions sous la forme matricielle : soit les éléments A et B sont indépendants, soit ils sont couplés.

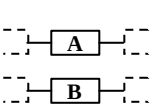
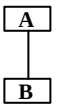
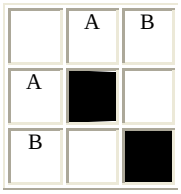
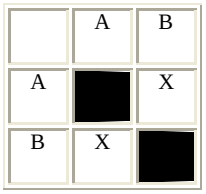
Matrices binaires symétriques		
Relation	**Non couplé**	**Couplé**
Graphe		
Matrice		

Tableau II-1. Caractéristiques des matrices binaires symétriques

- La deuxième classe correspond à des matrices non symétriques et regroupe les graphes orientés où les interactions peuvent être non symétriques et ainsi servir à modéliser des relations de hiérarchie, de contrainte et de précédence entre les éléments d'un système. Dans cette classe, trois blocs basiques (tableau II-2) permettent de modéliser les

interactions sous la forme matricielle : soit les éléments A et B sont indépendants et peuvent être réalisés en parallèle, soit l'un des éléments contraint l'autre (enchaînement séquentiel), soit les éléments se contraignent mutuellement (ils sont dits couplés).

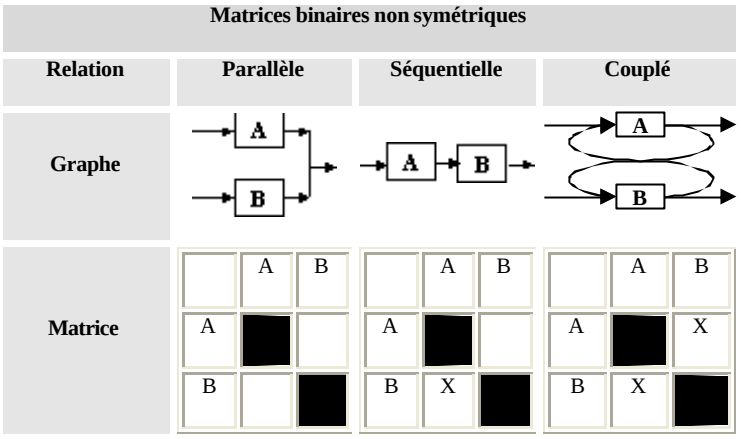


Tableau II-2. Caractéristiques des matrices binaires non symétriques

2.4. Les Matrices Structurelles de Conception : DSM

DSM est l'acronyme de « *Design Structure Matrix* » ce qui est traduit par matrice structurelle de conception. Dans la suite de ces travaux, nous allons utiliser le terme DSM pour faire référence à une matrice structurelle.

Une DSM est un outil de modélisation, utilisé dans les projets de conception et dérivé des matrices binaires avec l'introduction de nouvelles capacités et possibilités. Nous présentons dans ce qui suit les fondements de cet outil matriciel et ses apports dans la modélisation des domaines d'un projet.

2.4.1. Classification des DSM

Il existe deux classes de DSM, en lien avec la classification utilisée pour les matrices binaires. On retrouve (figure II-11) ainsi la classe des matrices symétriques sous le nom de DSM statiques et la classe des matrices asymétriques sous le nom de classe des DSM temporelles.

Les DSM statiques tirent leur nom du fait qu'elles sont utilisées pour matérialiser les interactions entre les éléments d'un système, indépendamment du temps. Les deux sous-classes les plus utilisées des DSM statiques sont les DSM Produit utilisées pour représenter l'architecture des produits et les DSM Organisation (ou Acteurs) pour modéliser l'organisation des acteurs dans un projet ou dans l'entreprise. Les DSM temporelles quant à elles permettent de représenter les systèmes dont les éléments sont liés par des relations de précédence, qui peuvent être des relations de précédence de type temporel (DSM Processus) ou des relations de hiérarchie ou de contrainte (DSM Paramètres).

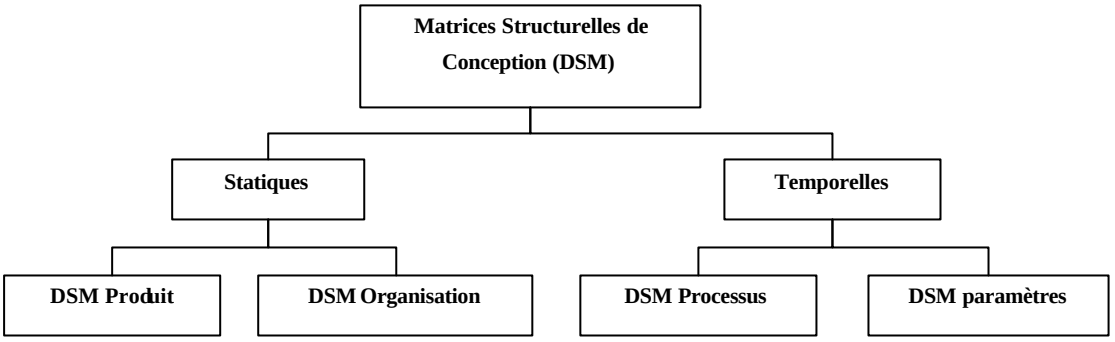


Figure II-11. Classification des DSM

2.4.2. Les DSM Temporelles

La terminologie exacte utilisée pour faire référence à la modélisation matricielle des systèmes hiérarchisés est « *Dependency Structure Matrix* », ce qui permet d'obtenir le même acronyme DSM.

Les deux sous-classes les plus connues des DSM temporelles sont les DSM processus et les DSM paramètres.

Les DSM Processus. Elles permettent de modéliser l'enchaînement des tâches de tous les processus de l'entreprise. Dans l'entreprise les tâches déterminent principalement l'allocation des ressources et servent de base pour estimer la durée et le coût du projet. C'est pourquoi les deux grands domaines d'application des DSM temporelles sont :

- L'optimisation des processus du point de vue coût et délais : nous citerons les travaux de Yassine et al. [2001] sur les probabilités de reconception, ceux de Carrascosa et al. [1998] sur l'estimation des délais de conception et ceux de Browning et d'Eppinger [2002] sur le lien entre le séquencement des tâches et les coûts de développement.
- L'optimisation des processus pour la conception des équipes de conception : nous citerons les travaux d'Eppinger et al. [1994], de Chen et Lin [2003], de Browning [1999] et de Braha [2002].

Les DSM Paramètres. Elles ont été initialement utilisées pour modéliser les relations de précédence et les contraintes entre les paramètres et variables en conception. Elles s'appliquent plus généralement à tous les systèmes dont les éléments ne sont pas homogènes.

L'objectif général lorsqu'on analyse des DSM temporelles est de réorganiser les éléments en ligne et en colonne de manière à réduire les retours en arrière lorsqu'il s'agit de tâches, ou d'identifier une hiérarchie entre les éléments. Sur la figure II-12, si on considère que les activités s'enchaînent de A vers K, alors on remarque qu'il y a trois retours en arrières (les croix au-dessus de la diagonale).

Cette réorganisation est assurée par des algorithmes de séquencement et de partitionnement.

ACTIVITES		A	B	C	D	E	F	G	H	I	J	K
Recevoir spécifications	A	A										
Générer concepts	B	X	B									
Concevoir les modèles	C	X	X	C								
Produire les modèles	D			X	D							
Développer programme de tests	E	X	X	X		E						
Tester les modèles	F			X	X	X	F					
Concevoir les plans	G	X	X	X			X	G	X	X		
Concevoir les moules	H	X	X				X	X	H	X		
Concevoir les outils d'assemblage	I							X	X	I		
Fabriquer les moules	J								X		J	
Initialiser la production	K											K

Figure II-12. Exemple de DSM temporelle

Le partitionnement est le processus de réorganisation des lignes et des colonnes d'une DSM temporelle de sorte qu'on réduise les interactions représentant les retours en arrière. Cela revient à rendre la DSM le plus possible triangulaire. Cependant cette action n'est pas toujours synonyme de réduction de la durée globale du processus modélisé. En effet, l'exemple en figure II-13 montre un processus composé de 7 tâches dont les relations de précédence sont modélisées par une DSM. La figure II-13(a) montre le résultat du partitionnement qui a pour but de réduire les boucles en arrière, on remarque qu'on a uniquement deux sauts en arrière de E vers A et de F vers C. On suppose que le processus en question est fini lorsqu'on passe au moins une fois par toutes les tâches et que toutes les tâches ont des durées équivalentes, alors on obtient l'enchaînement suivant (A, B, C, D, E, A, B, C, D, E, F, C, D, E, F, G) avec 16 tâches. On remarque alors, que plus les interactions sont éloignées de la diagonale, plus la boucle est grande.

Pour remédier à ces problèmes d'allongement des processus, une alternative est de réarranger les lignes et les colonnes des DSM temporelles de manière à ce que les interactions soient les plus proches de la diagonale. Ainsi, on augmente le nombre des boucles arrière, mais ces boucles seront plus courtes. La figure II-13(b) montre qu'avec la deuxième technique de séquencement, on obtient 4 boucles mais un processus plus court (A, E, A, E, D, E, D, B, D, B, C, F, C, F, G) avec 15 tâches.

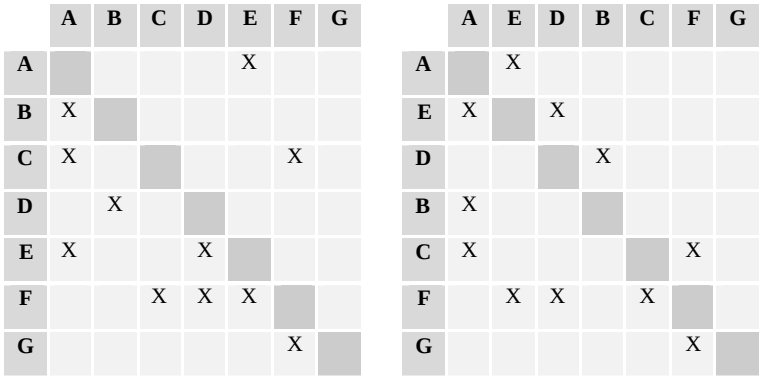


Figure II-13. Deux résultats de partionnement d'une DSM

2.4.3. Les DSM statiques

Les DSM statiques par opposition aux DSM temporelles permettent de modéliser d'une manière non hiérarchique les interactions entre les constituants d'un système, ce système pouvant être un produit ou une organisation.

Historiquement, les DSM statiques binaires ont été choisies [Kusiak et al., 1993] pour modéliser l'architecture des produits et modéliser les interactions qui lient leurs composants. Ces DSM sont binaires et permettent d'identifier l'existence ou non d'un type d'interaction étudiée. Les DSM statiques binaires sont encore utilisées malgré l'apparition des DSM numériques [Bohm et al., 2004].

Une DSM Produit modélise les interactions entre les constituants d'un produit. Pimmler et Eppinger [1994] proposent une taxonomie (tableau II-3) permettant de différencier les différents types d'interactions. Cette taxonomie permet ainsi de construire des DSM produits où les cellules, autres que les diagonales, contiennent des vecteurs de dimension 4 contenant des valeurs binaires.

Type de DSM	Type d'interaction	Explication
Produit	Spatiale	Lié à la propagation de contraintes physiques
	Energie	Lié aux flux d'énergie
	information	Lié au flux d'information
	matériel	Lié au flux de matière

Tableau II-3. Taxonomie des interactions du Produit [Pimmler et Eppinger, 1994]

Dans le cadre de l'IS, Meinadier [2002] justifie l'utilisation des matrices DSM auxquelles il donne le nom de matrices de couplage et d'intégration, car elles permettent en effet de représenter et visualiser facilement les interactions (de différentes natures) entre les constituants. Elles sont utiles d'abord lors de la définition de l'architecture organique (regroupement de modules, optimisation des couplages) et ensuite lors de l'intégration pour

vérifier que les zones de couplages et les interfaces entre constituants sont bien réalisées et vérifiées.

Avec une DSM « Organisation »[10], on modélise les interactions entre les acteurs dans une équipe de conception. Parmi les travaux les plus récents utilisant les DSM Organisation binaires, on citera ceux de [Sosa et al., 2004]. Dans ses travaux, Sosa propose la taxonomie des interactions présentée dans le tableau II-4.

Type de DSM	Type d'interaction	Explication
Acteurs	Qualité	Lié aux type de moyens de communication (direct, mail, document, etc.)
	Fréquence	Lié à la fréquence des interactions
	Direction	Lié à ses des interactions
	« Timing »	Lié à la référence des interactions par rapport au projet

Tableau II-4. Taxonomie des interactions entre acteurs [Sosa et al., 2004]

2.4.4. Synthèse sur les types de DSM

Le tableau II-5 résume les principaux types de DSM utilisées dans la modélisation de projet.

Type de DSM	Représentation	Application	Méthode d'analyse
Produit	Interactions entre composants	Conception et architecture des systèmes	Clustering
Acteurs	Caractérisation des interfaces entre acteurs	Conception d'organisation, gestion des interfaces entre acteurs, coopération et collaboration, flux d'information	Clustering
Activités	Relations Input/Output entre activités	Structuration des processus, planification des projets	partitionnement
Paramètres	Précédences et hiérarchies entre paramètres		partitionnement

Tableau II-5. Récapitulatif des caractéristiques des DSM

Avant la numérisation proprement dite des DSM, nous avons assisté à l'utilisation d'une forme intermédiaire entre les DSM binaires et celles numériques, il s'agit des DSM utilisant une métrique discrète la plupart du temps basée sur une sémantique linguistique du genre Faible, Moyen et Fort. Ainsi dans les travaux de Sosa [2005] on retrouve une métrique utilisant les classes Fort, Faible et Nul.

2.5. Les DSM numériques

Dans les DSM binaires, un seul attribut est utilisé pour caractériser les interactions à l'intérieur d'un système à savoir l'existence ou l'absence d'une interaction. Une quantification des interactions modélisées dans les DSM permet de relativiser le poids des interactions et ainsi d'augmenter la qualité informative des DSM.

[10] Terme anglo-saxon utilisé, nous le remplaçons par Acteurs dans notre travail

2.5.1. DSM temporelles numériques

Concernant les DSM temporelles, nous pouvons retenir les travaux d'Eppinger et al. [1994] qui proposent de rendre numérique une DSM processus (Figure II-4) en portant sur la diagonale la durée de chaque tâche et en transformant la représentation binaire des interactions en évaluation numérique du degré de dépendance entre les tâches (valeur comprise entre 0 et 1).

Le traitement proposé par les auteurs est de simuler le partitionnement qui réduit la durée totale du processus modélisé.

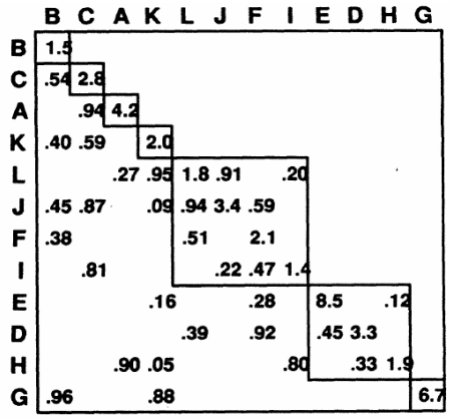


Figure II-14. Exemple de numérisation des DSM temporelles [Eppinger et al., 1994]

La question qu'on peut se poser lorsqu'on veut faire évoluer le caractère informatif des DSM binaires est : quels attributs rendre numériques ?

Dans la littérature, on trouve plusieurs méthodes de numérisation des DSM temporels. Dans ce qui suit, nous allons passer en revue les plus connues :

- Steward [1981] qui suggère d'utiliser des valeurs numériques pour caractériser l'ordre dans lequel les tâches de retour en arrière sont traitées. La marque avec la plus haute valeur est celle qui doit être analysée en premier.
- Yassine et al. [2001] qui proposent de quantifier le degré d'importance des interactions. Une échelle linguistique est utilisée pour caractériser les interactions : 1 = importante, 2 = moyenne, 3 = faible dépendance. Dans ce cas, on obtient trois classes de dépendance ordonnées selon leur degré d'importance.
- On peut caractériser la force du lien de dépendance entre les tâches, deux mesures sont usuellement utilisées soit entre 0 et 1 [Eppinger et al., 1994], soit entre 1 et 10 [Yassine et al., 2000]. L'algorithme de partitionnement utilisé a pour but alors de minimiser une somme globale des dépendances dans la DSM.
- On peut caractériser le volume des flux d'information. L'algorithme de partitionnement a les mêmes objectifs que celui du cas précédent [Guivarch, 2003].

- On peut caractériser la variabilité de l'information échangée. Cette mesure de variabilité reflète l'incertitude sur les échanges d'information entre tâches. Cette mesure peut être construite statistiquement ou évaluée subjectivement par les acteurs du projet [Yassine, 2004].
- On peut caractériser la probabilité de répétition, ainsi une valeur numérique renseigne sur la probabilité de causer des répétitions [Smith et al., 1997 ; Browning, 1998] [Cho et Eppinger, 2001]
- Carrascosa et al. [1998] proposent de construire et de caractériser les couplages entre tâches par des probabilités, puis de provoquer des changements, ce qui représente une vision plus large que le modèle présenté dans le point précédent.

Avant de définir les DSM statiques numériques, il est important d'ajouter que dans certains travaux, nous trouvons une modélisation du domaine des activités et des tâches par des DSM statiques numériques [Chen et Lin, 2003]. L'objectif est alors d'identifier les regroupements entre tâches pour les affecter par la suite à des acteurs collectifs (des équipes).

2.5.2. DSM statiques numériques

Les travaux d'enrichissement de l'outil DSM se sont intéressés tout d'abord aux DSM temporelles, ce qui a ouvert la porte à la numérisation des DSM statiques et plus spécifiquement à la numérisation des DSM Organisation. En effet, les premiers travaux sur la numérisation des DSM statiques ont été réalisés par McCord et Eppinger [1993] et ont porté sur la construction d'une DSM acteurs en partant des tâches que les acteurs réalisent. Ils proposent alors de quantifier les couplages entre les acteurs en utilisant une DSM temporelle numérique qui évalue l'intensité des dépendances entre les tâches.

La formalisation de la numérisation des DSM organisation a été réalisée quand les premières taxonomies des types d'interactions entre acteurs ont été proposées, ces taxonomies servaient de base pour la proposition de métriques. Morelli et al. [1995] ont utilisé la « qualité » comme métrique pour qualifier les moyens de communications entre les acteurs. Les travaux les plus récents quant à eux utilisent beaucoup plus la fréquence des interactions comme métrique [Sosa, 2004].

Quant aux DSM produit, certains travaux pionniers dans la numérisation [Pimmler et Eppinger, 1994] proposent une métrique discrète qui s'applique à tous les types d'interactions identifiées. Cette métrique représentée dans le tableau II-6 classe les interactions en 5 types. La DSM représentée dans la figure II-15 montre l'une des premières applications des DSM Produit numériques. Le produit en question est un moteur thermique.

Type de DSM	Type	Métrique	
Produit	Spatiale	Nécessaire : (+2)	L'interaction est nécessaire pour le fonctionnement du produit
	Energie	Désiré : (+1)	L'interaction est bénéfique mais non absolument nécessaire
	information	Indifférent : (0)	L'interaction n'affecte pas la fonctionnement du produit
	matériel	Non-désirée : (-1)	L'interaction affecte la fonctionnement du produit
		Nuisible : (-2)	L'interaction empêche le bon fonctionnement du produit

Tableau II-6. Métrique pour les DSM Produit Pimmler et Eppinger, 1994]

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Radiator A		2 0 0 2			2 -2 0 0											
Engine Fan B	2 0 0 2				2 0 0 2								1 0 0 0			
Heater Core C				1 0 0 0			2 0 0 0	-1 0 0 0								0 0 0 2
Heater Hoses D			1 0 0 0						-1 0 0 0							
Condenser E	2 -2 0 0	2 0 0 2				0 2 0 2		-2 2 0 2								
Compressor F					0 2 0 2			0 2 0 2	1 0 0 2	0 0 2 0	0 0 2 0		1 0 0 0			
Evaporator Case G			2 0 0 0					2 0 0 0						2 0 0 0	2 0 0 0	2 0 0 2
Evaporator Core H			-1 0 0 0		-2 2 0 2	0 2 0 2	2 0 0 0		1 0 0 2							0 0 0 2
Accumulator I				-1 0 0 0		1 0 0 2		1 0 0 2		1 0 0 0						
Refrigeration Controls J						0 0 2 0			1 0 0 0		0 0 2 0		1 0 0 0			
Air Controls K						0 0 2 0				0 0 2 0		0 0 2 0	1 0 0 0	0 0 2 0	0 0 2 0	
Sensors L											0 0 2 0		1 0 0 0			
Command Distribution M		1 0 0 0				1 0 0 0				1 0 0 0	1 0 0 0	1 0 0 0		1 0 0 0	1 0 0 0	1 0 0 0
Actuators N							2 0 0 0				0 0 2 0		1 0 0 0			
Blower Controller O							2 0 0 0				0 0 2 0		1 0 0 0			2 0 0 2
Blower Motor P			0 0 0 2				2 0 0 2	0 0 0 2					1 0 0 0		2 0 0 2	

Figure II-15. Exemple de DSM Produit [Pimmler et al., 1994]

Une autre métrique très répandue dans les DSM produit numériques, est celle qui utilise des valeurs numériques réelles la plupart du temps positives (bornées ou pas) [Larses, 2005] [Rodriguez-Torro, 2004] [Clarkson et al., 2004].

3. Modélisation par les DSM du produit et de l'organisation du projet

Dans notre démarche vers la proposition d'un outil d'identification des architectures du produit et de l'organisation du projet, nous proposons d'utiliser l'outil DSM et les MI pour modéliser respectivement les interactions inter-domaines et les interactions intra-domaines [Harmel et al., 2007]. Ce choix est justifié par le positionnement qu'on a fait concernant la caractérisation des architectures par les couplages.

La classification anglo-saxonne des DSM statiques en DSM Produit et DSM Acteurs ne reflète pas l'existence de deux formalismes différents mais seulement de deux cas d'application.

Dans notre travail, nous mettons beaucoup plus l'accent sur le fait que les DSM que nous utilisons sont des DSM statiques utilisées pour capturer à un instant donné les interactions et les couplages qui existent entre des éléments appartenant à un même domaine ou à des domaines différents.

Nous ferons remarquer au lecteur, comme nous l'avons souligné plus haut, que les DSM Processus peuvent avoir deux représentations possibles : l'une est la DSM temporelle qui est la plus usuellement utilisée. Dans ce cas, la finalité est d'identifier le meilleur séquencement des tâches pour satisfaire un objectif de réduction de durée, d'élimination de bouclage, etc… l'autre est la DSM statique. Notre choix s'est porté sur cette dernière possibilité.

Nous modélisons alors par ces outils matriciels les domaines du produit et de l'Organisation du projet (figure II-16). Concernant le Produit, nous définissons alors une DSM Fonctions Système (DSM FS) qui nous permettra d'identifier l'architecture fonctionnelle du produit, une DSM composant (DSM COMP) qui nous permettra d'identifier l'architecture organique du produit et une matrice d'incidence liant les Fonctions Système et les composants (MI FS-COMP) pour représenter les couplages entre les domaine fonctionnel et organique. Nous reproduisons cette modélisation pour l'organisation du projet.

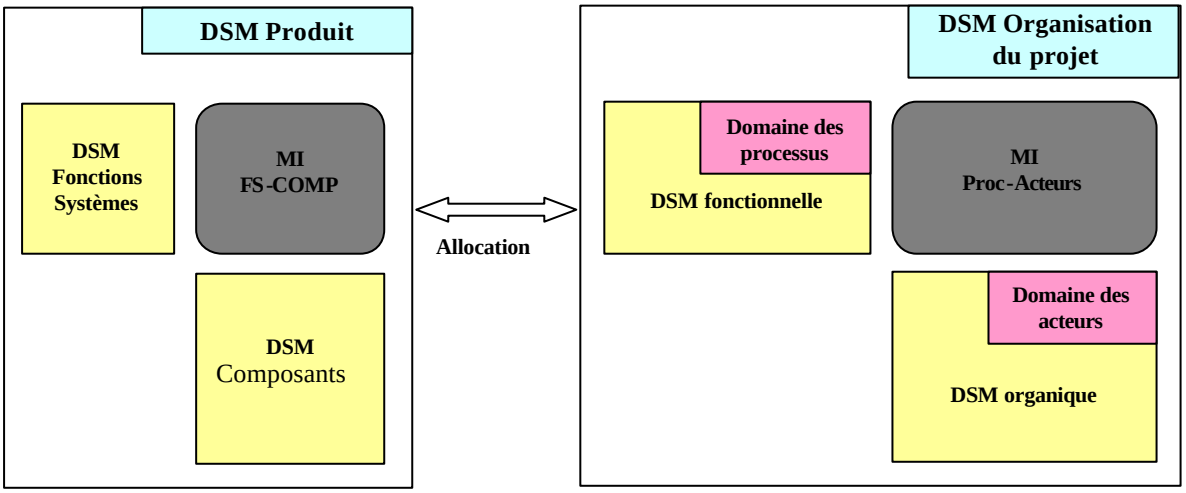


Figure II-16. Modélisation des domaines du produit et de l'organisation du projet

4. Synthèse

Dans ce chapitre, nous avons commencé par donner un aperçu sur les méthodes de représentation des architectures fonctionnelle et organique, nous avons ensuite approfondi la présentation des outils matriciels : matrice d'incidence et DSM.

En cohérence avec notre choix d'étudier les architectures des domaines du produit et de l'organisation du projet sur la base des couplages qui existent à l'intérieur et entre ces

domaines, nous avons opté pour l'utilisation des outils matriciels qui formalisent la représentation de ces interactions.

Nous introduisons dans le chapitre suivant les méthodes d'identification des architectures.

CHAPITRE III

CHAPITRE III

UN NOUVEL ALGORITHME DE CLUSTERING POUR L'IDENTIFICATION DES ARCHITECTURES

Dans ce chapitre, nous proposons un nouvel algorithme de *clustering*[11] qui utilise le formalisme de représentation matricielle pour mettre en œuvre l'identification des architectures des domaines du projet. Le principe retenu est d'utiliser les DSM en entrée pour proposer des architectures cohérentes avec les définitions retenues des concepts de modules et d'éléments intégrateurs, inspirées de la typologie d'Ulrich [1995].

Notre contribution concerne l'amélioration d'un algorithme de clustering existant qui a servi de référence à plusieurs travaux antérieurs. Dans ce chapitre, nous faisons une synthèse sur les différents types d'algorithmes qui existent et en particulier, sur ceux qui utilisent comme donnée une DSM. Nous présentons brièvement d'autres types d'algorithmes tels que les Algorithme Génétiques (AG). Ensuite, nous détaillons l'algorithme qui a servi de base et de référence à notre travail. Nous proposons alors plusieurs améliorations de cet algorithme, à partir de notre analyse bibliographique sur la conception modulaire et les indices de modularité. Enfin, nous comparons l'algorithme amélioré à l'algorithme de départ en utilisant comme critères la reproductibilité et la pertinence des résultats, en soulignant ainsi les gains obtenus avec l'algorithme proposé.

[11] Nous utiliserons le terme anglo-saxon "clustering" qui signifie regroupement.

1. Sur les algorithmes de clustering

Le « *clustering* »[12] fait référence au classement d'objets dans différents groupes, plus précisément au partionnement des données en paquets homogènes qu'on appelle « cluster »[13]. Les objets appartenant à un cluster partagent des caractéristiques communes, qui correspondent le plus souvent à des critères de proximité que l'on définit en introduisant des mesures de distance. Selon Hartigan [1975], « le but originel du clustering est de trouver des similitudes entre éléments afin de les grouper ensemble en se basant sur un seuil de ressemblance ».

Il y a plusieurs méthodes pour mettre en œuvre le clustering, par exemple : les coefficients de similarité [Bezdek et Pal, 1992], le tri, l'optimisation de coûts [Idicula, 1995 ; Fernandez, 1998 ; Thebeau, 2001], l'itération et les algorithmes génétiques [Kusiak et al., 1993 ; Rogers et McCulley, 1996 ; Yu et al. 2003].

On classe les algorithmes de clustering en algorithmes hiérarchiques ou de partionnement. Les algorithmes hiérarchiques identifient les clusters en utilisant ceux qui sont identifiés à l'étape précédente. Les algorithmes de partionnement trouvent directement tous les clusters, mais pour ces algorithmes, le nombre de clusters est fixé (méthode K-means).

Les algorithmes hiérarchiques peuvent procéder de deux manières différentes : soit par des agrégations successives, c'est la méthode Bottom-Up, soit par division, c'est la méthode Top-Down.

Toutes ces méthodes nécessitent des mesures de distance ou de ressemblance pour caractériser les clusters. Ces métriques se basent sur les attributs des objets pour proposer une « fonction objectif »[14] de la distance. Plusieurs travaux sur les métriques de distances et les algorithmes de clustering existent, les pionniers étant Alexander [1964] et Hartigan [1975].

Dans ce travail, nous n'allons pas approfondir l'étude des métriques et des fonctions objectives qui définissent les algorithmes de clustering et ce pour deux raisons principales : d'abord à cause de leur grand nombre, ensuite à cause de leur diversité liée aux types de problème, à leur complexité et aux types d'algorithme de clustering.

[12] http://en.wikipedia.org

[13] Nous utiliserons dans la suite de ce document les termes "cluster" et "module" comme synonymes mais nous privilégierons le terme francophone.

[14] En recherche opérationnelle, le terme retenu est aussi "fonction coût". Il s'agit de la fonction qu'il s'agit d'optimiser.

1.1. Le clustering des DSM

La plupart des algorithmes de clustering n'utilisent pas les DSM comme moyen de représentation des données, mais plutôt les graphes et les hypergraphes. Cependant, les DSM offrent principalement deux avantages :

- Les DSM sont modélisées par une forme mathématique (les matrices) qui facilite la mise œuvre des algorithmes de clustering.
- Les DSM facilitent la mise en œuvre d'une méthode de modularisation cohérente avec notre définition d'un module (pour rappel, un module est un ensemble d'éléments fortement liés entre eux et faiblement liés à d'autres éléments externes au module).

Les DSM sur lesquelles on peut appliquer des algorithmes de clustering sont les DSM statiques. Les travaux relatifs au clustering des DSM statiques sont dans une grand majorité issus du MIT (*Massachusetts Institute of Technology*) et plus précisément des départements d'ingénierie mécanique et du centre international pour la recherche sur le management des technologies, ce dernier réalisant le transfert des technologies issues du MIT vers l'industrie. Les premiers travaux sur le clustering des DSM se situent en 1995 quand Idicula [Idicula, 1995] a soutenu une thèse, encadré par Eppinger. L'algorithme de clustering développés par Idicula a été ensuite repris et amélioré successivement par Fernandez [1998] et Thebeau [2001]. L'algorithme proposé dans ces travaux repose sur une méthode hiérarchique Bottom-Up par optimisation de coût.

D'autres méthodes de clustering des DSM existent. La plus connue après la méthode d'optimisation de coût est le clustering par des algorithmes génétiques [Yu et al., 2003] [Whitfield et al., 2002].

La méthode de clustering adoptée dans ce travail s'appuie sur l'algorithme initialement développé par Idicula. Avant d'approfondir cette méthode de clustering, nous allons présenter succinctement l'autre méthode de clustering, à savoir le clustering par des algorithmes génétiques.

1.2. Le clustering des DSM par les algorithmes génétiques

En dépit du grand nombre d'applications des algorithmes génétiques (AG) dans les problèmes d'optimisation, les travaux de recherche utilisant les AG dans les problèmes de clustering des DSM sont peu nombreux.

Des travaux récents existent pour le traitement des DSM temporelles, dont l'objectif est d'optimiser le séquencement et l'enchaînement des tâches [Altus et al., 1996], [McCulley et al., 1996].

L'utilisation des algorithmes génétiques dans les problèmes de clustering des DSM est encore plus récente. Nous pouvons mentionner les travaux de [Demeriz et al., 1999], de [Whitefield

et al., 2002] et de [Yu et al., 2003]. L'algorithme utilisé dans ces deux derniers travaux est celui de Goldberg [1989]. Cependant plusieurs modifications ont été introduites, que ce soit au niveau des fonctions objectives ou sur les procédures régissant le fonctionnement de l'AG.

Il faut noter cependant que Whitfield n'a pas utilisé directement l'AG pour proposer des architectures modulaires mais pour tester ces architectures en adoptant la métrique MSI (paragraphe 3.8 du chapitre I). Concernant Yu, il a utilisé un formalisme complexe qui est très peu référencé dans notre domaine. Ce formalisme est connu sous le non de « longueur minimale de description » (en Anglais, Minimum Description Length) dont la publication de référence est [Rissien, 1978]. D'un autre côté, l'AG de Yu a été utilisé uniquement sur des DSM binaires avec des contraintes bien spécifiques et il n'y a pas, à notre connaissance, de travaux plus récents permettant l'extension des AG aux DSM numériques.

1.3. Synthèse sur les algorithmes de clustering

Notre principal objectif n'était pas le développement d'un algorithme de clustering mais d'une méthode d'identification des architectures des domaines du projet. Dans un premier temps, nous avons donc opté pour l'utilisation d'un algorithme de référence, connu pour donner de bons résultats : l'algorithme d'Idicula [Idicula, 1995]. De plus, cet algorithme a déjà fait l'objet d'analyses et d'études contradictoires, qui ont mis en évidence ses avantages et ses limites. Par la suite, après une période d'utilisation de cet algorithme et après analyse des indices de modularité proposés dans la littérature, nous avons remarqué que des améliorations significatives de cet algorithme étaient possibles.

La partie suivante (partie 2.) présente l'algorithme qui nous sert de référence. Ensuite, nous proposerons une adaptation de cet algorithme (partie 3.) ainsi qu'une comparaison mettant en évidence l'amélioration obtenue.

2. L'algorithme de référence de Idicula (1995)

Idicula [1995] s'est intéressé à deux problèmes rencontrés en ingénierie concourante et liés aux tâches de conception. Le premier problème était « d'identifier l'ensemble de tâches interdépendantes dans un processus de conception ». Le second avait pour sujet « la détermination des groupements des tâches interdépendantes ». L'algorithme proposé par Idicula pour le deuxième problème est un algorithme de clustering hiérarchique du type Bottom-Up qui groupe « les tâches de conception dans des modules qui sont faiblement couplés entre eux, alors que chaque module est composé de tâches fortement couplées ». Notons qu'Idicula s'intéressait aux couplages entre tâches (et aux coûts de coordination associés) mais sans introduire l'aspect temporel : il traitait donc bien des DSM statiques.

Pour identifier ces modules, Idicula a utilisé un algorithme stochastique qui recherche itérativement à réduire la valeur d'une « fonction objectif » qu'il appelle « coût total de

coordination ». L'algorithme peut être configuré de différentes façons pour produire différents groupements.

2.1. La méthodologie de l'algorithme

Nous allons présenter maintenant la méthodologie de l'algorithme dans sa dernière version avec les améliorations apportées par Fernandez [1998] et Thebeau [2001]. Précisons que Fernandez et Thebeau ont travaillé respectivement sur des DSM acteurs et des DSM composants mais ils ont conservé la même dénomination de la « fonction objectif », c'est-à-dire coût total de coordination. Dans la présentation de l'algorithme développé par les chercheurs du MIT, nous conserverons leurs notations.

La méthodologie de l'algorithme peut être résumée de la façon suivante (Figure III-1) :

- En premier, faire de chaque élément de la DSM un module et calculer un coût total de coordination.
- Ensuite, un élément est choisi aléatoirement et il est assigné à un module qui a présenté la plus forte enchère pour prendre cet élément. Si le fait de créer ce nouveau module réduit le coût total de couplage, alors le module est accepté. Les modules vides, les modules en doublon et les modules contenus entièrement dans d'autres modules sont supprimés.
- L'algorithme répète ce processus jusqu'à ce qu'on n'obtient plus aucune amélioration du coût total de coordination.
- Il y a plusieurs dispositifs aléatoires implémentés dans l'algorithme qui permettent la formation de modules, même s'ils ne présentent pas « les interactions les plus fortes » ou une amélioration du coût de coordination. Cet aspect aléatoire permet à l'algorithme de couvrir au maximum l'espace des solutions admissibles, de sortir d'optima locaux et ainsi d'obtenir les solutions finales avec le coût total de coordination « le plus bas possible »[15].

Un organigramme de l'algorithme est présenté dans la figure III-1. Les caractéristiques les plus importantes de l'algorithme sont incluses dans l'organigramme.

[15] Cette exploration de l'espace des solutions ne peut pas garantir l'atteinte d'un optimum mais la procédure du recuit simulé, que nous détaillerons par la suite, est réputée pour permettre de trouver de bonnes solutions, avec des temps de calcul limités.

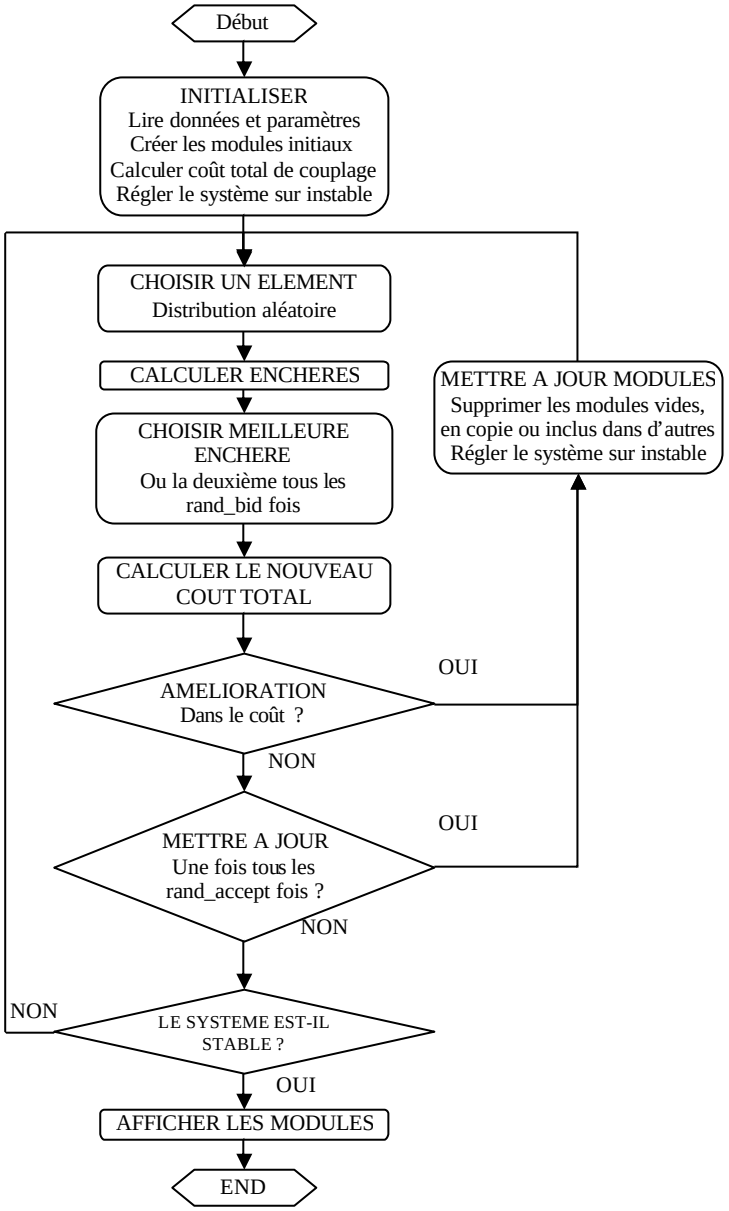


Figure III-1. Organigramme de l'algorithme [Fernandez, 1998]

Il y a quatre points importants qui font la spécificité de cet algorithme :

- La fonction d'enchère qui permet à un module de tester le coût de l'incorporation d'un élément ;
- La « fonction objectif » qui exprime le coût de coordination de l'architecture testée et qui garantit l'obtention de l'architecture la plus adaptée en minimisant ce coût ;
- Le recuit simulé qui permet à l'algorithme de sortir du chemin d'un optimum local pour trouver une architecture qui minimise le coût de coordination ;
- Il permet aussi à un élément d'appartenir à plusieurs modules et s'oppose de ce fait à la pratique commune de ne faire appartenir un composant qu'à un seul et unique module.

2.1.1. La fonction d'enchère

L'algorithme choisit aléatoirement et donc équiprobablement un élément. Une fois que l'élément est choisi, l'algorithme calcule une enchère de chaque module. Une telle enchère est une mesure de la façon dont les membres « forts » d'un module interagissent avec l'élément choisi. L'équation III-1 montre la forme exacte de la fonction d'enchère.

$$Bid\left(cluster_k, element_i\right) = \sum_{j=1}^{size} \frac{\left(\left(DSM(i,j) + DSM(j,i)\right)^{pow\_dep} \times CL\_MAT(k,j)\right)}{cl\_size(k)^{pow\_bid}} \quad \textbf{Eq. III-1}$$

Avec:

$Bid\left(cluster_k, element_i\right)$	*Est l'enchère du module k envers l'élément i.*
$DSM(i,j)$	*Est la valeur de l'interaction entre i et j dans la matrice.*
size	*Est la taille de la DSM, càd le nombre d'éléments composant le domaine.*
$cl\_size(k)$	*Est le nombre d'éléments contenus dans le module k.*
pow_dep	*Contrôle l'importance donnée aux interactions fortes relativement aux interactions faibles. Une grande valeur augmente cette différence.*
pow_bid	*Contrôle la valeur de l'enchère relativement à la taille des modules. Une grande valeur décourage la formation de grands modules et encourage la formation de modules de la même taille.*
$CL\_MAT(k,j)$	*C'est une variable binaire. Il prend 1 quand l'élément j est dans le module k.*

2.1.2. La fonction Coût

La fonction « coût total de coordination » est l'agrégation des coûts de coordination de toutes les tâches. Dans l'algorithme de référence, l'auteur distingue deux situations différentes pour le calcul du coût de coordination entre les tâches.

Première situation : si les tâche i et j interagissent et si cette interaction est incluse dans un module – autrement dit, i et j appartiennent au même module, c'est alors l'équation III-2 qui s'applique pour calculer le coût de coordination [16]:

$$Coût\ de\ Coordination\left(Tâche_i, Tâche_j\right) = \left(DSM(i,j) + DSM(j,i)\right) \times \left(cluster(i,j)^{pow\_cc}\right) \quad \textbf{Eq.III-2}$$

Seconde situation : si les tâches i et j n'appartiennent pas au même module, c'est alors l'équation III-3 qui s'applique :

$$Coût\ de\ Coordination\left(Tâche_i, Tâche_j\right) = \left(DSM(i,j) + DSM(j,i)\right) \times \left(size^{pow\_cc}\right) \quad \textbf{Eq.III-3}$$

[16] Nous conservons ici la dénomination utilisée par les chercheurs du MIT.

On peut vérifier qu'une interaction nulle induit un coût de coordination nul. Le coût total de coordination est alors la somme des coûts de coordination de toutes les interactions (Eq.III-4) :

$$Coût\ Total\ de\ Coordination = \sum_{i=1}^{size}\sum_{j=1}^{size} Coût\ de\ Coordination\left(Tâche_i, Tâche_j\right) \quad \textbf{Eq.III-4}$$

Avec :

$cluster(i, j)$	*Est la taille du module qui contient i et j .*
pow_cc	*Est le paramètre qui contrôle la pénalité assignée à la taille des modules dans le coût.*

L'algorithme d'Idicula sélectionne d'une manière aléatoire une tâche et calcule des enchères pour les différents modules. L'enchère la plus élevée est choisie, la tâche est alors associée à ce module et on calcule le nouveau coût total de coordination. S'il y a amélioration du coût alors la tâche est incluse dans ce module. Ce processus se poursuit d'une manière itérative et se termine lorsqu'aucune amélioration du coût de coordination ne survient, pendant un « certain » nombre d'itérations (paramètre à fixer).

2.1.3. Le recuit simulé

Dans deux étapes de l'algorithme, la décision portant sur le choix de l'étape suivante n'est pas déterminée par les résultats obtenus mais par un processus aléatoire. Ce type de processus est connu sous le nom de « Simulated Annealing » qui en français est traduit en « Recuit Simulé ».

Le Recuit Simulé est un méta-algorithme probabiliste générique, il permet de réaliser une bonne approximation de l'optimum global d'une « fonction objectif » dans un grand espace de définition. Il a été développé parallèlement par Kirkpatrick, Gelatt et Vecchi [Kirkpatrick et al., 1983], et par Cerny [1985].

La dénomination de cet algorithme est inspirée de la métallurgie, où on répète cycliquement la chauffe et le refroidissement d'un métal pour augmenter la taille de ses cristaux et réduire ses défauts.

Le principe de l'algorithme est le suivant : l'algorithme est initialisé avec une solution prise au hasard. À chaque itération, il modifie légèrement la solution pour essayer d'améliorer la « fonction objectif ». Des sauts aléatoires sont introduits pour permettre de sortir d'optima locaux, et donc accepter des solutions même si elles n'améliorent pas la « fonction objectif ». Des conditions d'arrêt de l'algorithme sont définies pour limiter la durée d'exploration de l'espace de solutions. La solution obtenue est généralement de bonne qualité (voire optimale), même si l'utilisateur n'a pas de garantie d'optimalité.

En incorporant ce méta-algorithme dans un algorithme de clustering, nous créons une divergence dans le processus de décision qui permet à l'algorithme d'atteindre des solutions qu'il n'aurait pas pu atteindre normalement.

Les paramètres *rand_bid* et *rand_accept* contrôlent ces deux étapes dans l'algorithme. Le premier affecte la probabilité de choix de la meilleure enchère et laisse la possibilité à d'autres modules de tenter d'optimiser la « fonction objectif ». Le paramètre *rand_accept* quant à lui contrôle la probabilité qu'un module soit modifié bien qu'aucun changement dans la « fonction objectif » ne soit réalisé. Dans les deux situations, ce processus de saut introduit des modifications aléatoires dans la progression de l'algorithme, c'est-à-dire dans la recherche de la meilleure solution. De plus, ces sauts nous garantissent l'exploration par l'algorithme d'un espace de solutions plus grand, ce qui évite d'être piégé dans une solution localement optimale.

La figure III-2 montre l'historique des coûts calculés à chaque boucle de l'algorithme.

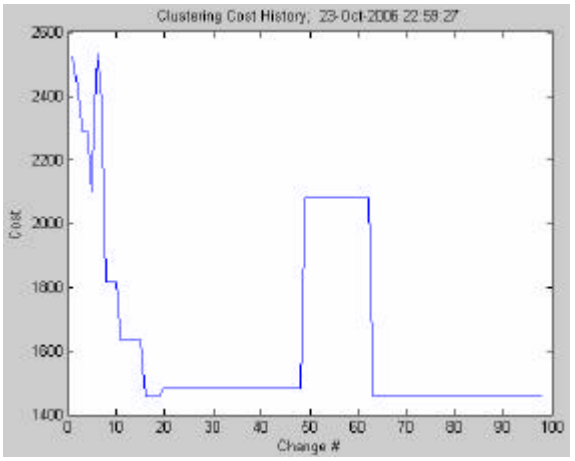


Figure III-2. Exemple de l'historique du Coût Total

Nous remarquons à travers cette figure que le recuit simulé a permis de relancer l'algorithme trois fois vers des architectures qui n'améliorent pas le coût. Ceci a eu pour résultat d'aboutir à une architecture qui a un plus faible coût que les autres optima locaux.

Après avoir résumé la méthodologie de lalgorithme de référence et ses points forts, nous allons présenter dans le paragraphe suivant les paramètres qui permettent de contrôler le fonctionnement de l'algorithme.

2.2. Les données d'entrée et les paramètres

Les données et les paramètres passés à l'algorithme déterminent le comportement de l'algorithme et le type de résultat obtenu. Une DSM numérique, listant les interactions entre les éléments d'un domaine et évaluant l'intensité de ces interactions, structure nos données d'entrée. Les paramètres déterminent la façon dont l'algorithme explore et trouve une solution.

Les auteurs successifs qui ont travaillé sur cet algorithme font remarquer que l'intensité relative des interactions peut influencer le résultat du clustering, spécialement pour les cas où la métriques utilisée est discrète et repose sur deux ou trois valeurs possibles.

Par exemple s'il y a seulement deux niveaux d'interaction : Fort et Faible, l'utilisateur devrait décider si deux interactions faibles additionnées ensemble sont plus, moins ou aussi importantes qu'une unique interaction forte. Ainsi, différents cas de figure sont possibles sur l'importance relative des intensités. Par exemple, cas 1 : Faible=3, Fort=10 ; cas 2 : Faible=5, Fort=10 ; cas 3 : Faible=7, Fort=10. Les auteurs précisent alors que pour une même DSM, avec ces trois configurations, on peut obtenir trois architectures différentes. Il y a huit paramètres différents employés pour commander et contrôler l'algorithme. Ce sont tous des nombres entiers positifs. Ici nous présenterons brièvement ce qu'ils contrôlent et comment ils peuvent être utilisés pour influencer les résultats de l'algorithme. Le choix de la valeur de certains paramètres peut « favoriser » l'obtention d'un certain type de solution, ce qu'il est nécessaire de rappeler ici. Nous rappelons dans le tableau III-1, les paramètres passés à l'algorithme.

size	*Est la taille de la DSM, càd le nombre d'éléments composant le domaine.*
pow_cc	*Contrôle la pénalité assignée à la taille du module dans le coût de couplage.*
pow_bid	*Est semblable au précédent, sauf qu'il est utilisé dans la fonction d'enchère qui sera expliquée ultérieurement. Les valeurs usuelles sont entre 0 et 3.*
pow_dep	*Est aussi utilisé dans la fonction d'enchère. Les valeurs usuelles sont entre 0 et 2, les valeurs les plus élevées mettent en avant les interactions fortes.*
max_Cl_size	*Empêche la formation de modules contenant plus que max_Cl_size éléments. Si l'utilisateur désire restreindre le nombre d'éléments dans un module, ce paramètre doit être ajusté à la valeur désiré.*
rand_accept	*Impose à l'algorithme de faire un changement tous les « rand_accept » itérations, même si aucune amélioration n'a été obtenue dans le coût de couplage. On a utilisé des valeurs entre 0.5*size et 2*size.*
rand_bid	*Impose à l'algorithme de prendre la seconde plus haute enchère chaque « rand_bid » itérations. On a utilisé des valeurs entre 0.5*size et 2*size.*
times	*Détermine le nombre de fois (en fait, (times×size)) que l'algorithme essaiera de sélectionner un élément et de créer un module avant de vérifier la stabilité du système.*
stable_limit	*Détermine une condition d'arrêt de l'algorithme. (stable_limit ´ (times ´ size)) itérations pourront être réalisées sans amélioration dans le coût de couplage avant de prendre la décision de stopper l'algorithme.*

Tableau III-1. Les paramètres de l'algorithme initial

Le tableau III-2 donne un aperçu des valeurs utilisées par Fernandez [1998] et Thebeau [2001] dans leurs applications.

Paramètres	Fernandez	Thebeau
size	23	61
pow_cc	2	1
pow_bid	2	1
pow_dep	2	4
max_Cl_size	23	61
rand_accept	23	122
rand_bid	10	122
times	2	2
stable_limit	2	2

Tableau III-2. Initialisation des paramètres dans l'algorithme original

Le paramètre **pow_cc** pénalise la taille du module dans le calcul de coût. On a remarqué que les valeurs plus grandes que 2 ne modifient pas le clustering obtenu.

Le paramètre **pow_bid** est initialement fixé à zéro ce qui a pour conséquence de ne pas pénaliser le module à cause de sa taille dans le processus d'enchère. Une valeur non nulle favorise les modules avec une grande taille dans le calcul des enchères.

Le paramètre **pow_dep** permet de favoriser les interactions fortes durant le processus d'enchère.

La partie « simulated annealing » de l'algorithme de clustering utilise les paramètres **rand_bid** et **rand accept,** qui spécifient le nombre de fois que l'algorithme pourra faire un changement non optimal. Des résultats expérimentaux ont permis de mettre en évidence que des valeurs égales à la dimension du domaine (*size*) permettent d'obtenir des résultats satisfaisants.

Concernant les paramètres **times** et **stable_times**, elles ont été fixées à 2. L'expérience a montré que ces valeurs sont suffisantes pour obtenir les meilleurs résultats.

2.3. Simulation de l'architecture optimale

L'algorithme de clustering présenté dans cette partie est un algorithme hiérarchique du type bottom-up. Il procède par agrégations successives pour identifier l'architecture optimale. Cependant, cet algorithme ne permet pas les retours en arrière, dans le sens où on ne peut pas défaire un module pour en créer un autre. Il peut dupliquer un élément et le faire appartenir à plusieurs modules, si cela présente un avantage pour le coût total. On peut enregistrer les architectures simulées, pour retenir la meilleure à la fin de l'algorithme.

Cette caractéristique fait que l'algorithme ne suit pas d'une manière déterministe le chemin qui mène à la solution optimale. De ce fait, il peut proposer des architectures non optimales.

Pour remédier à ce problème, les auteurs proposent de simuler plusieurs fois le clustering d'une DSM donnée. On obtient alors avec des fréquences variables (selon la taille et la densité de la DSM) une architecture minimisant le coût total, parmi d'autres dont les coûts sont supérieurs.

Il n'est pas possible de vérifier si l'architecture avec le coût minimal est celle qui réalise effectivement l'optimum global du problème. En effet, si on considère une DSM de dimension 13 avec trois modules (l'arrangement des éléments dans un module n'affecte pas le coût) de taille 6, 4 et 3 alors il y a 60060 architectures possibles. Cependant, le nombre de modules n'étant ni fixe, ni connu à l'avance, on a alors comme valeur supérieure du nombre d'architectures possibles le nombre de permutations. Ce nombre dépasse les 9 milliards.

Dans la suite de nos travaux à chaque fois que nous ferons référence, *par abus de langage*, à l'architecture optimale, au coût optimal ou à la solution optimale, il faut comprendre que nous faisons référence à la solution proposée par l'algorithme et non à la solution optimale du problème.

3. Proposition d'un nouvel algorithme de clustering

Dans ce qui suit, nous allons présenter l'algorithme de clustering amélioré et son fonctionnement.

L'algorithme utilisé se base sur l'évaluation des couplages entre éléments d'un même domaine pour générer d'une manière systématique des groupements en modules de ces mêmes éléments en optimisant une « fonction objectif ». Il ne gère que les valeurs positives des couplages.

Nous avons réutilisé la méthodologie de l'algorithme d'Idicula, mais nous avons opté pour une dénomination plus générique de la « fonction objectif » indépendante du domaine étudié, à savoir : « coût total de couplage ».

L'algorithme d'Idicula permet de produire d'une manière stochastique des modules à partir d'une DSM. Cependant, cet algorithme est lourd à utiliser et les paramètres de contrôle sont trop nombreux pour permettre de trouver une combinaison optimale.

Dans notre démarche d'amélioration de l'algorithme de référence, nous avons intégré des notions fondamentales de l'architecture modulaire afin d'orienter le fonctionnement de l'algorithme.

Les modifications que nous avons apportées concernent cinq points spécifiques :

- La fonction d'enchère,
- La « fonction objectif »,

- Le recuit simulé,
- L'identification des éléments intégrateurs,
- Et les paramètres de fonctionnement.

3.1. La fonction d'enchère

La fonction d'enchère a un rôle très important dans l'optimisation de l'algorithme de clustering que nous proposons. C'est la fonction d'enchère qui nous permet d'associer à un élément un module bien précis qui peut accepter favorablement ce nouvel élément. On peut aisément comprendre que sans la fonction d'enchère, l'algorithme de clustering devient un algorithme d'exploration aléatoire.

En effet, après la sélection aléatoire d'un élément ($élément_i$), la fonction d'enchère (que nous appelons *enchère*, *Cf* Eq. III-5) évalue pour chaque module M_j l'intérêt d'intégrer cet élément. Le module qui propose la meilleure enchère est présélectionné et le nouveau coût total est alors calculé. Nous avons adopté la fonction d'enchère suivante :

$$enchère\ \left(M_j, élément_i\right) = CM\left(\ \{\ M_j; élément_i\ \}\ \right) - CM\left(M_j\right) \qquad \textbf{Eq.III-5}$$

Avec CM, une fonction qui calcule la Cohésion d'un Module donné. Cette fonction est proche des mesures IM et ARP utilisées par [Blackenfelt, 2000]. Cette formulation permet de favoriser la mise en place d'une architecture modulaire.

L'expression de CM est donnée dans l'équation III-6.

$$CM\left(M_k\right) = \frac{\left(\sum_{i \in M_k} \sum_{j \in M_k} \left(\ DSM(i,j) + DSM(j,i)\ \right)\right)^{\exp\_\text{int}}}{\left(\ taille\left(M_k\right)\ \right)^{\exp\_taille}} \qquad \textbf{Eq.III-6}$$

Avec

taille(M)	*Est une fonction qui donne le nombre d'éléments qui compose un module donné M. Cette fonction était appelée cl_size dans l'algorithme initial.*
DSM(i,j)	*Est la valeur du couplage entre l'élément i et l'élément j. A noter : quand i=j, DSM(i,i)=0, car la diagonale de la DSM n'a aucun sens.*
exp_int	*Est un paramètre qui permet d'ajuster l'importance des interactions au sein d'un module dans la fonction d'enchère*
exp_taille	*Est un paramètre qui permet d'ajuster l'importance de la taille des modules dans la fonction d'enchère.*

Afin d'expliquer le comportement de la fonction d'enchère, nous allons considérer un exemple didactique. Nous considérons une DSM de dimension 6, décrite dans la figure III-3. L'architecture représentée correspond à l'étape où les modules I=(A, B, C) et II=(D, E) vont

proposer une enchère pour l'élément F. Nous avons fixé arbitrairement, pour cet exemple, exp_int à 2 et exp_taille à 1.

	A	B	C	D	E	F
A		5	4			2
B	5		3			
C	4	3				5
D					5	1
E				5		7
F	2		5	1	7	

Figure III-3. Situation initiale avant enchère

	A	B	C	F	D	E
A		5	4	2		
B	5		3			
C	4	3		5		
F	2		5		1	7
D				1		5
E				7	5	

	A	B	C	F	D	E
A		5	4	2		
B	5		3			
C	4	3		5		
F	2		5		1	7
D				1		5
E				7	5	

Figure III-4. Architecture visée pour chaque enchère

On obtient $enchère\ (I,F)=169$ et $enchère\ (II,F)=175.33$. Donc c'est le module II qui remporte les enchères. Ainsi la fonction d'enchère agit comme une mesure de l'évolution de la densité d'un module entre l'avant et l'après de l'incorporation d'un nouvel élément.

La fonction d'enchère proposée ne prend en compte que les interactions internes au module, elle favorise donc la création de modules qui présentent une forte cohésion entre les éléments qui les composent.

3.2. La fonction Coût Total de Couplage

L'algorithme est construit autour d'une « fonction objectif » appelée le coût total de couplage. Cette fonction a pour but de faire correspondre notre vision de l'architecture d'un système à la formulation mathématique qu'elle utilise. Ainsi, le but est de transcrire les observations suivantes dans une formulation mathématique :

- Le temps ou le coût de traitement d'une interaction est proportionnel à l'importance de cette interaction. Une interaction importante ou plus fréquente exige plus d'attention, plus de ressources, ou plus de travail des concepteurs. Par conséquent, un couplage avec une valeur plus élevée aura un coût de couplage plus élevé.
- Nous supposons ici que dans une architecture modulaire, un élément appartenant à un module est fortement couplé aux autres éléments de ce module, alors qu'au même

moment, cet élément est faiblement couplé à ceux appartenant à d'autres modules. D'ailleurs, un élément intégrateur est un élément qui ne peut appartenir à aucun module puisqu'il est couplé à beaucoup d'éléments appartenant à différents modules.

- Lorsque la taille d'un module est importante, il devient difficile et coûteux à concevoir à cause de sa complexité. De plus, il est moins aisé à standardiser. Nous considérons alors que le coût de couplage d'une interaction est une fonction croissante de la taille du module qui la contient.

- Lorsqu'une interaction lie deux éléments appartenant à deux modules différents alors le coût de spécification/définition de cette interaction dépend de la taille des deux modules qu'elle rapproche. Cette remarque est à l'origine de l'évolution du coût de couplage que nous proposons par la suite (*Cf* Eq.III-8).

Pour obtenir l'expression finale du coût total de couplage, nous avons procédé en deux temps :

- Dans un premier temps, à l'image de la proposition d'Idicula [1995], nous avons construit deux fonctions complémentaires du coût de couplage : l'une pour caractériser les interactions internes à un module et l'autre pour caractériser les interactions externes à ce module.

- Dans un second temps, nous avons essayé d'améliorer l'efficacité des ces fonctions en les couplant à des métriques de modularité.

Pour chaque interaction dans la matrice DSM, l'algorithme initial calcule un coût de couplage. Ensuite, la somme de tous les coûts de couplage donne le coût total de couplage. Les équations Eq.III-7 et Eq.III-8 montrent les coûts de couplage pour une interaction.

Si les deux éléments i et j appartiennent au même module k (M_k), alors :

$$Coût\ Couplage\ (i,j) = (DSM(i,j) + DSM(j,i)) \times (taille(M_k))^2 \qquad \textbf{Eq.III-7}$$

Sinon, si aucun module ne contient les éléments i et j, alors il existe deux modules $M_{i'}$ et $M_{j'}$ qui contiennent respectivement les éléments i et j. Le coût de l'interaction entre les éléments i et j s'écrit alors :

$$Coût\ Couplage\ (i,j) = (DSM(i,j) + DSM(j,i)) \times (taille(DSM) + taille(M_{i'}) + taille(M_{j'}))^2 \qquad \textbf{Eq.III-8}$$

L'équation III-9 résume l'expression du coût total de couplage. C'est cette fonction que l'algorithme a pour objectif de minimiser.

$$Coût\ Total\ Couplage = \sum_i \sum_j Coût\ Couplage\ (i,j) \qquad \textbf{Eq.III-9}$$

Les trois formulations précédentes des coûts de couplage peuvent être réécrites en adoptant comme référence les modules. On peut formuler alors une fonction de coût de couplage, soit interne, soit externe à un module k, M_k (Eq.III-10 et III-11).

$$Coût\ Couplage\ Interne(M_k) = \sum_{i \in M_k} \sum_{j \in M_k, j \neq i} (DSM(i,j) + DSM(j,i)) \times \left(taille(M_k) \right)^2 \quad \textbf{Eq.III-10}$$

$$Coût\ Couplage\ externe(M_k) = \sum_{i \in M_k} \sum_{j \notin M_k, j \in M_m} \left[(DSM(i,j) + DSM(j,i)) \times (taille(DSM) + taille(M_k) + taille(M_m))^2 \right]$$

Eq.III-11

Alors le Coût Total de Couplage peut s'écrire sous la forme présenté dans l'équation III-12.

$$Coût\ Total\ Couplage = \sum_{M_k}^{Modules} (Coût\ Couplage\ Interne(M_k) + Coût\ Couplage\ Externe(M_k))$$

Eq.III-12

Dans un second temps, nous avons voulu améliorer la fonction de coût proposée pour mettre l'accent sur deux caractéristiques d'une architecture modulaire :

- Un module est le groupement d'éléments interagissant fortement entre eux
- Les éléments appartenant à un module interagissent faiblement avec les éléments appartenant à d'autres modules.

A la lumière de ces deux remarques, nous avons opté pour une adaptation de l'indicateur MSI proposé par [Whitfield et al., 2002] (*page 44*). L'indicateur MSI se compose de deux parties, MSIi et MSIe.

La partie MSIi (pour indicateur MSI interne) prend en compte les interactions internes à un module et donne de ce fait une mesure de la cohésion et de la robustesse (Strength) interne du module.

La partie MSIe (pour indicateur MSI externe) prend en compte les interactions externes qui lient les éléments appartenant au module aux autres éléments externes au module, ainsi MSIe mesure la force des interactions externe au module.

Il apparaît à travers les deux définitions que MSIi et MSIe sont deux indicateurs antagonistes. C'est pourquoi l'indicateur MSI s'écrit comme la différence des deux.

Dans notre travail, nous n'allons pas utiliser directement l'indicateur MSI, mais une forme dérivée des deux indicateurs MSIi et MSIe. Alors que ces deux indicateurs originaux tiennent compte de l'intensité des interactions, les indicateurs que nous proposons tiennent compte uniquement de l'existence ou non des interactions. On obtient alors les indicateurs MSIi' et MSIe' présentés en Equation III-13 et III-14.

$$MSI_i'(M_k) = \frac{\sum_{i=n_1}^{n_2} \sum_{j=n_1, j\neq i}^{n_2} d(i,j)}{(n_2 - n_1 + 1)^2 - (n_2 - n_1 + 1)} \quad \textbf{Eq.III-13}$$

$$MSI_e'(M_k) = \frac{\sum_{i=1}^{n_1} \sum_{j=n_1}^{n_2} 2d(i,j)}{2 \times (n_1 \times (n_2 - n_1 + 1))} + \frac{\sum_{i=n_2}^{taille(DSM)} \sum_{j=n_1}^{n_2} 2d(i,j)}{2 \times ((taille(DSM) - n_2) \times (n_2 - n_1 + 1))} \quad \textbf{Eq.III-14}$$

Avec :

$d(i,j)$	*Prend la valeur 1 quand* $DSM(i,j) \neq 0$ *et la valeur 0 quand* $DSM(i,j) = 0$.
n_1	*Index du premier élément du module* M_k
n_2	*Index du dernier élément du module* M_k ($n_2 \geq n_1$). *Si* $n_2 = n_1$, *le module ne contient en fait qu'un seul élément.*

Notons que le dénominateur correspond au nombre total d'interactions possibles, en dehors de la diagonale et que nous considérons uniquement des DSM symétriques.

L'indicateur MSI a été développé pour des DSM numériques qui prennent leurs valeurs entre 0 et 1. 0n obtient alors des indicateurs MSIi et MSIe ayant leurs valeurs entre 0 et 1. Du fait que les indicateurs MSIi' et MSIe' prennent en compte des interactions valant 1 ou 0, MSIi' et MSIe' ont une valeur bornée par 0 et 1.

Avant d'expliquer l'utilisation de ces indicateurs dans la fonction de coût, nous allons étudier le comportement de ces deux indicateurs.

L'indicateur MSIi' est maximal et vaut 1, lorsque le module est totalement dense, c'est à diretous les éléments qui le composent interagissent. L'indicateur est nul, lorsque les éléments qui composent le module n'interagissent pas du tout entre eux (ce qui remet en question l'utilisation du terme module pour qualifier ce groupement d'éléments).

L'indicateur MSIe' est nul lorsque le module concerné n'interagit avec aucun élément externe (il est complètement découplé). Il est maximal lorsque chacun des éléments qui le composent interagit avec tous les éléments externes à ce module. Cette dernière configuration est également contraire à l'application du terme module à un groupement d'éléments.

Ainsi, un groupement d'éléments tend à devenir un module parfait quand son MSIi' tend vers 1 et son MSIe' tend vers 0.

Si nous reprenons la « fonction objectif » choisie plus haut (Eq.III-12), nous remarquons que nous pénalisons plus fortement les interactions externes que les interactions internes. Cette configuration peut aboutir selon l'exemple traité à la création de modules relativement grands, même si la taille du module est pénalisée. Ainsi, pour ajuster et corriger la « fonction

objectif », nous adoptons l'une et/ou l'autre des corrections présentées en Equation III-15 et III-16.

$$Coût\ Couplage\ Interne\ Corrigé(M_k) = Coût\ Couplage\ Interne\ (M_k) / MSIi(M_k) \quad \textbf{Eq.III-15}$$

$$Coût\ Couplage\ Externe\ Corrigé(M_k) = Coût\ Couplage\ Externe(M_k) \times MSIe(M_k) \quad \textbf{Eq.III-16}$$

Ces deux corrections sont interprétées de la manière suivante :

- En divisant le coût de couplage interne par MSIi', nous favorisons la création de modules plutôt denses que grands ;
- En multipliant le coût de couplage externe par MSIe', nous favorisons la minimisation des interactions externes aux modules.

Le Coût Total Corrigé s'écrit alors :

$$Coût\ Total\ Couplage\ Corrigé = \sum_{M_k}^{Modules} (Coût\ Couplage\ Interne\ Corrigé(M_k) + Coût\ Couplage\ Externe\ Corrigé(M_k))$$

Eq.III-17

Afin de montrer la pertinence de cette dernière « fonction objectif », nous allons comparer dans le dernier paragraphe de ce chapitre notre algorithme et l'algorithme référence.

3.3. Le recuit simulé

Dans l'ultime évolution de l'algorithme d'Idicula, proposée par Thebeau [2001], le recuit simulé permet à l'algorithme de réaliser aléatoirement certaines opérations, ce qui a pour intérêt d'augmenter le nombre de solutions explorées par l'algorithme. Cependant, nous avons remarqué une erreur liée à l'utilisation du recuit simulé.

Cette erreur se situe dans la boucle qui permet d'accepter un coût de couplage supérieur, l'algorithme de référence enregistre alors la meilleure architecture référencée en cours obtenue jusqu'à là (*best_curr_cost)* et qui peut être obtenue dans la boucle de stabilité en cours, mais cette architecture n'est pas sauvegardée dans la variable *best_coord_cost* qui contient la meilleure architecture obtenue. Il s'ensuit à la sortie de cette boucle que l'algorithme peut accepter un coût qui améliore *best_coord_cost* sans vérifier si *best_curr_cost* améliore *best_coord_cost*. Par conséquent, l'algorithme peut ne pas prendre en compte la meilleure architecture parmi celles qu'il teste. Pour remédier à cette erreur, nous avons vérifié dans un premier temps, que *best_curr_cost* peut être meilleur (plus petit) que *best_coord_cost*. Ensuite, nous avons ajouté une boucle de test qui compare les deux solutions et qui enregistre la meilleure des deux solutions.

3.4. Identification des éléments intégrateurs

Si on prend en référence uniquement les interactions qui lient les éléments entre eux, un élément intégrateur est en premier lieu un élément qui n'appartient à aucun module et qui interagit avec des éléments appartenant à plusieurs modules.

Cependant l'objectif principal de l'algorithme de *clustering* de référence est d'agréger les éléments pour construire des modules. D'un autre côté, la situation de départ pour l'algorithme est celle où chaque élément forme un module, c'est à direoù chaque élément est considéré comme étant intégrateur. Cette configuration est non souhaitée et a de ce fait un coût important. On arrive alors au constat que d'abord il est difficile de caractériser les éléments intégrateurs et qu'ensuite l'algorithme favorise l'intégration des éléments pour former des modules.

Nous distinguons deux types d'éléments intégrateurs :

- le premier type correspond aux éléments intégrateurs qui interagissent fortement et en nombre avec plusieurs éléments du système. Par exemple, l'élément H dans la DSM de la Figure III-5(a) est clairement un élément intégrateur.
- Le deuxième type d'éléments intégrateurs comprend ceux qui ont peu d'interactions. Par exemple, l'élément H de la figure III-5(b) qui a moins d'interactions par comparaison à celui de la première DSM, mais peut encore être considéré comme intégrateur.

	A	B	C	D	E	F	G	H
A		10	8					4
B	10		6					6
C	8	6						10
D					10			2
E				10				8
F							8	10
G						8		4
H	4	6	10	2	8	10	4	

(a) H : Premier type d'éléments intégrateur

	A	B	C	D	E	F	G	H
A		10	8					
B	10		6					
C	8	6						10
F					10			
D				10				8
E							8	10
G						8		
H			10		8	10		

(b) H : deuxième type d'élément intégrateur

Figure III-5. Les deux types d'éléments intégrateurs

Avec la « fonction objectif » que nous avons construite, il est possible de détecter automatiquement le premier type d'éléments intégrateurs. Cependant, il est presque impossible que le deuxième type puisse être identifié.

Ainsi, pour pallier ces incertitudes et étant donné la difficulté à déterminer certains éléments intégrateurs, nous avons introduit deux procédures pour donner la possibilité à l'utilisateur d'identifier le caractère intégrateur d'un élément :

- Premièrement, pour optimiser l'identification des éléments intégrateurs du premier type, nous introduisons un Indice de Couplage (IC). L'IC mesure le taux de couplage entre un élément et le reste du système. Par exemple, un élément avec un IC de 80% est en interaction avec 80% des autres éléments du système (l'équation dd montre l'expression mathématique de l'IC). Ainsi, nous permettons à l'utilisateur de préciser un seuil pour IC. Au-delà de ce seuil, les éléments sont considérés comme étant intégrateurs.

$$IC(élément_j) = \frac{\sum_{i=1, i\neq j}^{taille(DSM)} d(i, j)}{taille(DSM) - 1} \qquad \textbf{Eq.III-18}$$

- Deuxièmement, pour le deuxième type d'éléments intégrateurs, nous permettons à l'utilisateur de choisir directement les éléments qu'il considère comme étant intégrateurs.

L'exemple suivant permet de juger de l'utilité de ces deux procédures. Considérons la DSM d'un système composé de 12 éléments, décrite sur la figure III-6 (a). L'architecture présentée est celle qui est attendue par les architectes système. Elle correspond à un Coût de Couplage Total de 4513. On remarque qu'il y a trois modules et deux éléments intégrateurs. Le premier élément intégrateur K a 4 interactions, il ne se distingue donc pas des autres éléments. Le deuxième élément intégrateur L a quant à lui 7 interactions dont 4 avec le premier module (A,B,C,D,E).

	A	B	C	D	E	F	G	H	I	J	K	L
A		1	1		1							1
B	1		1	1							1	1
C	1	1		1		1						
D		1	1		1							1
E	1		1	1								1
F			1				1	1				1
G						1		1			1	
H						1	1				1	
I										1		1
J									1		1	
K		1					1	1		1		1
L	1	1		1	1	1			1		1	

(a) avec les procédures d'identification des éléments intégrateurs

	A	B	C	D	E	L	F	G	H	K	I	J
A		1	1		1	1						
B	1		1	1		1				1		
C	1	1		1			1					
D		1	1		1	1						
E	1		1	1		1						
L	1	1		1	1		1			1	1	
F			1			1		1	1			
G							1		1	1		
H							1	1		1		
K		1				1		1	1			1
I						1						1
J										1	1	

(b) sans les procédures

Figure III-6. Exemple d'architecture avec des éléments intégrateurs

En appliquant l'algorithme de clustering sans les procédures spécifiques aux éléments intégrateurs, nous obtenons l'architecture présentée dans la figure III-6(b). Cette architecture montre que l'algorithme a intégré l'élément L dans le premier module ce qui a permis d'éliminer quatre interactions externes et a intégré l'élément K dans le deuxième module avec lequel il interagissait prioritairement. Objectivement, l'architecture obtenue est meilleure que la première du point de vue « mesure de la modularité » avec un coût de 2073. Cet exemple

montre que l'algorithme initial n'est pas conçu pour identifier des éléments intégrateurs tels que K et L.

Pour tenir compte du jugement des architectes, nous fixons IC à 60% (ce qui permet automatiquement d'identifier L comme étant intégrateur) et nous classons manuellement K comme élément intégrateur. Avec cette semi-automatisation du clustering, on obtient alors l'architecture attendue, représentée en figure III-6 (a). Nous pouvons remarquer que l'élément K est couplé avec chacun des modules existants, ce qui justifie bien son caractère intégrateur. Nous reviendrons ultérieurement sur la difficulté intrinsèque à tout algorithme de clustering pour identifier "automatiquement" les éléments intégrateurs.

3.5. Les paramètres de fonctionnement

L'algorithme de référence peut être ajusté par 8 paramètres. Ces paramètres sont présentés dans le paragraphe 2.2 de ce chapitre.

Le tableau III-3 présente la liste des paramètres originaux et ceux que nous avons retenus pour notre algorithme.

Nom du paramètre initial	Nom du paramètre que nous retenons
pow_cc	Non utilisé
pow_bid	exp_taille
pow_dep	exp_int
max_Cl_size	Non utilisé
rand_accept	Pris égal à la taille de la DSM
rand_bid	Pris égal à la taille de la DSM
times	« qualité »
stable_limit	

Tableau III-3. Comparaison des paramètres dans les deux algorithmes

Nous remarquons à travers le tableau précédent que le nombre de paramètres que nous retenons passe de 8 à 5 dont deux pris égaux à la taille de la DSM. Nous allons expliquer les raisons de ces choix :

- pow_cc n'a pas d'équivalent dans notre algorithme. Après différents tests, nous avons fixé la pénalisation des tailles des modules à 2, valeur qui nous a permis d'obtenir de bons résultats ;
- pow_bid devient exp_taille. Ce paramètre permet d'ajuster le poids de la taille des modules dans la fonction d'enchère. Nous utilisons pour exp_taille les valeurs de 0, 1 ou 2. La valeur doit être choisie relativement à celle donnée à exp_int.
- pow_dep devient exp_int. Ce paramètre permet d'ajuster le poids de l'intensité des interactions dans la fonction d'enchère. Nous utilisons pour exp_int les valeurs de 0, 1 ou 2. La valeur doit être choisie relativement à celle donnée à exp_taille.

- max_Cl_size n'est pas utilisé, la taille des modules est limitée par la prise en compte de la taille des modules dans le calcul des coûts et par l'indicateur SMIi' qui favorise des modules denses.
- La pratique et les travaux antérieurs de Fernandez et Thebeau nous ont montré que les paramètres rand_accept et rand_bid ne doivent pas être très grands (ne pas faire de recuit simulé), ou très petits (ne pas laisser l'algorithme poursuivre un chemin cohérent). De ce fait une valeur égale à la taille de la DSM donne de bons résultats.
- Les paramètres times et stable_limit permettent d'ajuster le nombre de boucle que fait l'algorithme dans son travail d'exploration. La pratique nous a montré qu'une valeur de 2 suffit pour des problèmes de taille inférieure à 20, et qu'une valeur de 4 donne généralement de bons résultats avec toutes les tailles. Nous avons ainsi fusionné deux paramètres initiaux dans le paramètre « qualité » que nous utilisons généralement entre 2 et 6.

3.6. Comparaison à l'algorithme de référence

Dans cette partie, nous comparons l'algorithme de clustering que nous avons développé à la dernière évolution de l'algorithme d'Idicula [1995], à savoir l'algorithme de Thebeau [2001] qui est aussi le plus performant.

Notre algorithme reprend la même architecture que l'algorithme de Thebeau (enchère, coût total, recuit simulé). Mais il se différencie avec une nouvelle fonction d'enchère, une nouvelle fonction de coût total et une procédure nouvelle pour l'identification des éléments intégrateurs.

La comparaison entre les deux algorithmes se fera en deux temps. Premièrement, nous allons comparer l'efficacité des nouvelles fonctions d'enchère et de coût total. Pour cela, nous testerons les deux algorithmes sur des architectures uniquement modulaires sans éléments intégrateurs. Nous proposons 5 exemples de DSM dont les dimensions vont de 8 à 16. Ensuite, nous allons comparer les deux algorithmes sur un exemple introduisant des éléments intégrateurs.

3.6.1. Critère et test d'efficacité

Les algorithmes ont été configurés comme représenté dans le tableau III-4.

Paramètres	Valeurs pour notre algorithme	Valeurs pour l'algorithme initial
pow_cc	Non utilisé	2
exp_taille	1	1
exp_int	2	2
max_Cl_size	Non utilisé	Taille(DSM)
rand_accept	Taille(DSM)	Taille(DSM)
rand_bid	Taille(DSM)	Taille(DSM)
times	4	4
stable_limit	Non utilisé	4

Tableau III-4. Configuration des deux algorithmes

Afin de tester l'efficacité des deux algorithmes, nous devons retenir un ou des critères de comparaison. Il est d'abord important de noter que les coûts obtenus ne sont pas comparables puisque la méthodologie utilisée par les deux algorithmes est semblable mais diffère par les fonctions d'enchère et de coût total. Le but des adaptations que nous avons réalisées est de simplifier la configuration de l'algorithme et d'améliorer la densité des modules. Il faut donc pouvoir comparer les algorithmes sur des exemples avec une architecture de référence et nous pourrions évaluer leur capacité à converger vers cette architecture. Nous avons donc retenu comme critère, leur reproductibilité que nous mesurons par la fréquence d'obtention de la meilleure solution connue. Nous avons réalisé 40 simulations et ainsi généré 40 architectures pour chaque exemple. Nous avons observé, dans chaque exemple, que la meilleure architecture identifiée est la même pour les deux algorithmes. Cependant, la fréquence est plus forte avec notre algorithme.

Nous avons généré manuellement des DSM de taille différente, en leur attribuant des caractéristiques particulières et en construisant manuellement l'architecture qui serait attendue par l'architecte. Cette architecture servira de base pour comparer les deux algorithmes.

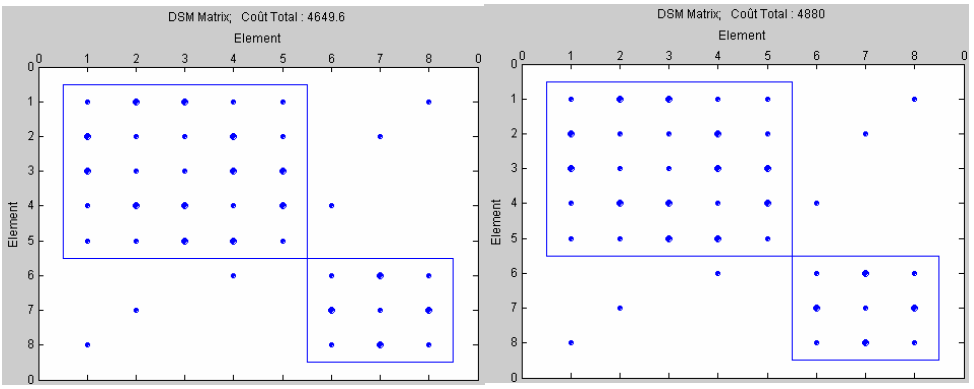
3.6.1.1. DSM de dimension 8

La figure III-7 montre à la fois la DSM de dimension 8 que nous avons utilisée en entrée des deux algorithmes et l'architecture optimale attendue. Cette DSM est de densité égale à 57%.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1		7	8	6	5			3
2	7		5	7	6		4	
3	8	5		8	7	2		
4	6	7	8		8			
5	5	6	7	8				
6				2			8	6
7		4				8		7
8	3					6	7	

Figure III-7. DSM de dimension 8 et l'architecture attendue

Les deux algorithmes identifient cette architecture (Figure III-8).



(a) Notre algorithme (b) Algorithme de Thebeau

Figure III-8. Architectures optimales pour les deux algorithmes

Le tableau III-5 résume la fréquence d'obtention de l'architecture optimale, on remarque que les deux algorithmes ont des fréquences proches avec un léger avantage pour l'algorithme original.

	Notre algorithme	**Algorithme original**
Fréquence	31/40=77%	35/40=87%

Tableau III-5. Fréquence de l'architecture optimale pour les deux algorithmes

3.6.1.2. DSM de dimension 10

La figure III-9 montre la DSM de dimension 10 que nous avons utilisée en entrée des deux algorithmes et l'architecture attendue. Par rapport à la DSM précédente, nous avons bien sûr augmenté la taille de la DSM, mais nous avons aussi introduit plus d'interactions externes et des interactions nulles à l'intérieur des modules. La densité globale est de 51%.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1		8	0	7	6	5			3	
2	8		8	5	6	7				4
3	0	8		5	0	8	2			
4	7	5	5		7	6		7		
5	6	6	0	7		7				
6	5	7	8	6	7					
7			2					8	7	6
8					7		8		4	5
9	3						7	4		8
10		4					6	5	8	

Figure III-9. DSM de dimension 10 et l'architecture attendue

Le tableau III-6 résume la fréquence d'obtention de l'architecture optimale, on remarque que les deux algorithmes ont des fréquences proches avec un léger avantage pour notre algorithme. Nous remarquons que les deux fréquences sont élevées.

	Notre algorithme	Algorithme original
Fréquence	36/40=90%	34/40=85%

Tableau III-6. Fréquence de l'architecture optimale pour les deux algorithmes

3.6.1.3. DSM de dimension 12

La figure III-10 présente la DSM de dimension 12 que nous avons testée en entrée des deux algorithmes. Par rapport à la DSM précédente, nous avons augmenté la taille de la DSM, mais nous avons aussi diminué la densité des modules et nous proposons des interactions externes multiples pour les éléments 3 et 8. La densité globale est de 48%.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1		8	0	7	6	5	4					
2	8		0	8	7	6	5					8
3	0	0		8	5	6	7			5	7	
4	7	8	4		8	0	7		4			
5	6	7	5	8		4	0	2				
6	5	6	6	0	4		6	6				
7	4	5	7	7	0	6						5
8					2	6			0	8	7	6
9				4				0		7	6	0
10			5					8	7		5	4
11			7					7	6	5		8
12		8						6	0	4	8	

Figure III-10. DSM de dimension 12 et l'architecture attendue

Le tableau III-7 résume la fréquence d'obtention de l'architecture optimale, on remarque que l'évolution de l'algorithme que nous proposons se détache avec un écart de 15%. Nous remarquons que dans cet exemple, notre algorithme prend un léger avantage.

	Notre algorithme	Algorithme original
Fréquence	33/40=82%	27/40=67%

Tableau III-7. Fréquence de l'architecture optimale pour les deux algorithmes

3.6.1.4. DSM de dimension 14

La figure III-11 montre la DSM de dimension 14 que nous avons utilisée en entrée des deux algorithmes, ainsi que l'architecture attendue. Dans cette DSM, nous avons augmenté la densité des interactions externes et diminuer la densité des modules. La densité globale est de 39,5%.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1		8	0	5	4	6	5						8	
2	8		6	2	0	5		8				6		
3	0	6		0	7	9					3	2		
4	5	2	0		3	7				4				
5	4	0	7	3		5							2	2
6	6	5	9	7	5				4	3				
7	5							8	0	2	7	3		
8		8					8		8	6	0		5	
9						4	0	8		7	0			
10				4		3	2	6	7		9			
11			3				7	0	0	9				3
12		6	2				3						4	5
13	8				2			5				4		6
14					2						3	5	6	

Figure III-11. DSM de dimension 14 et l'architecture attendue

Les deux algorithmes identifient comme optimale l'architecture attendue, mais cette fois-ci les fréquences chutent considérablement. Pour notre algorithme, cette architecture reste la première du point de vue de la fréquence d'obtention (ici, de 25 %). En ce qui concerne l'algorithme initial, il aboutit plus fréquemment vers une autre architecture (avec une fréquence de l'ordre de 28%) qui n'est pas la meilleure et dans 20% des cas, il converge vers l'architecture attendue. Ceci nous amène à supposer que les extremums locaux sont peut-être plus nombreux.

	Notre algorithme	**Algorithme original**
Fréquence	10/40=25%	8/40=20%

Tableau III-8. Fréquence de l'architecture optimale pour les deux algorithmes

3.6.1.5. DSM de dimension 16

La figure III-12 montre la DSM de dimension 16 utilisée en entrée des deux algorithmes, ainsi que l'architecture attendue. Les interactions externes sont relativement denses avec une densité de 27% et les modules ont une densité de 70%. La densité globale quant à elle est de 42.5%.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		7	8	6	0	5	4		2		3		5			
2	7		0	5	6	7	8	1		4		2		3		
3	8	0		6	8	6	0								3	3
4	6	5	6		9	0	8		2	3		1				
5	0	6	8	9		7	6	4			2					
6	5	7	6	0	7		0		1			3				
7	4	8	0	8	6	0							1	1		
8		1			4				4	6	7	8		2		1
9	2			2		1		4		0	8	7			3	
10		4		3				6	0		0	9				
11	3				2			7	8	0		0	2			
12		2		1		3		8	7	9	0					4
13	5						1				2			8	4	2
14		3					1	2					8		0	8
15			3						3				4	0		7
16			3					1				4	2	8	7	

Figure III-12. DSM de dimension 16 et architecture attendue

Le tableau III-9 résume la fréquence d'obtention de l'architecture optimale, on remarque que les fréquences s'établissent aux alentours de 75% pour notre algorithme contre 57% pour l'algorithme original.

	Notre algorithme	**Algorithme original**
Fréquence	30/40=75%	23/40=57%

Tableau III-9. Fréquence de l'architecture optimale pour les deux algorithmes

3.6.2. Test de pertinence

Dans ce paragraphe, nous comparons la pertinence des deux algorithmes dans l'identification de l'architecture attendue ou optimale. Nous avons identifié deux facteurs qui peuvent influencer l'obtention de l'architecture optimale. Le premier est la densité de la matrice. Le deuxième est la valeur relative des interactions externes et internes pour un module donné.

3.6.2.1. Influence de la densité

Considérons la DSM de dimension 16 représentée dans la figure III-13. Cette DSM a été construite sur la base de celle qui a été représentée en figure III-12, mais avec une densité globale plus forte (46%) et surtout une densité des interactions externes aux modules attendue de l'ordre de 36%. Nous constatons qu'après clustering, les interactions externes ont des valeurs inférieures à 4, tandis que les interactions internes aux modules sont supérieures à 4.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		0	8	7	0	7	6		4		3		5			
2	0		0	9	6	7	8	3		4		4		3		
3	8	0		6	8	9	0			5					3	2
4	6	5	6		0	0	8		4	3		3				
5	0	6	8	0		7	9	4			4				4	
6	5	7	6	0	7		0		3			3				3
7	4	8	0	8	6	0					4		3	3		
8		3			4				9	6	7	8		4		3
9	4			4		3		9		0	8	7			3	
10		4	5	3				6	0		0	9	3	3		
11	3				4		4	7	8	0		0	4			
12		4		3		3		8	7	9	0				3	4
13	5						3			3	4			8	9	0
14		3					3	4		3			8		0	8
15			3		4				3			3	9	0		8
16			2			3		3				4	0	8	8	

Figure III-13. DSM de dimension 16 et l'architecture attendue

Avec notre algorithme, nous obtenons comme architecture optimale, l'architecture attendue (Figure III-14). Ainsi la densité croissante des interactions externes n'a pas influencé le résultat. Nous pouvons remarquer que ce sont les interactions fortes à l'intérieur du module qui créent la cohésion.

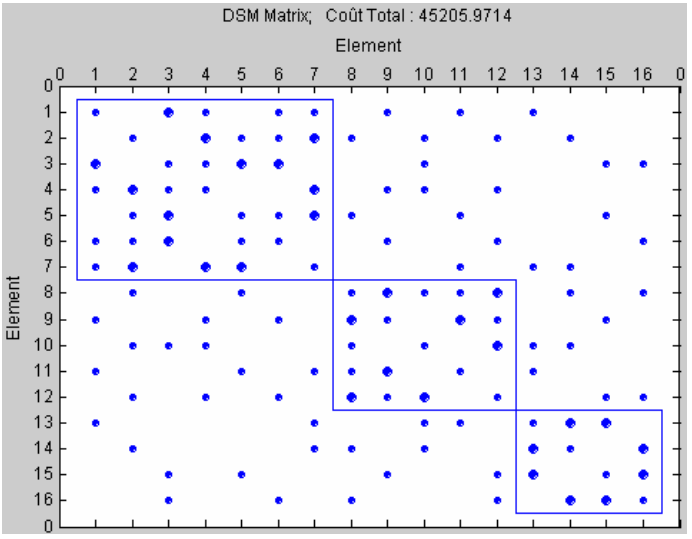


Figure III-14. Architecture optimale pour notre algorithme

L'algorithme de Thebeau propose l'architecture représentée sur la figure III-15. L'architecture se compose de deux modules de tailles 9 et 7. Nous remarquons que les densités des modules sont relativement faibles, (respectivement 52% et 66%) par comparaison aux 66%, 80% et 66% de l'architecture attendue. Mais la taille des modules obtenus fait que les interactions externes sont moins nombreuses. Nous avons lancé l'algorithme initial à 40 reprises. Il n'a jamais atteint l'architecture optimale.

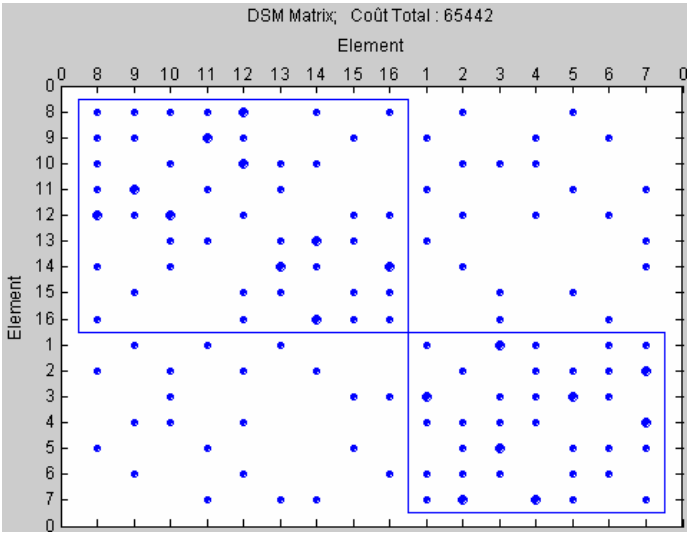


Figure III-15. Architecture optimale pour l'algorithme de Thebeau

Ainsi avec deux fonctions de coût dont une qui pénalise plus fortement les interactions externes, l'algorithme de Thebeau tend naturellement à créer de grands modules car cela permet de réduire fortement les interactions externes.

En raison des deux indicateurs que nous avons introduits et plus précisément dans ce cas avec MSI_i' notre algorithme limite la taille des modules en imposant une contrainte de densité.

3.6.2.2. Influence de l'intensité relative des interactions

Considérons la DSM représentée sur la figure III-16, elle ressemble fortement à celle en représenté en figure III-9 sauf que toutes les interactions internes ont une intensité égale à 3 et les interactions externes sont égales à 9.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1		3	3	3	3	3			9	
2	3		3	3	3	3				9
3	3	3		3	3	3	9			
4	3	3	3		3	3		9		
5	3	3	3	3		3				
6	3	3	3	3	3					
7			9					3	3	3
8					9		3		3	3
9	9						3	3		3
10		9					3	3	3	

Figure III-16. DSM de dimension 10 et l'architecture attendue

En utilisant les deux algorithmes de clustering avec cette DSM, nous obtenons deux architectures optimales différentes, et ce, lors du lancement de 40 simulations avec chacun des algorithmes. La figure III-17 montre l'architecture obtenue par notre algorithme. Il s'agit bien de l'architecture attendue avec deux modules complètement denses et des interactions externes limitées en nombre, même si elles sont de forte intensité.

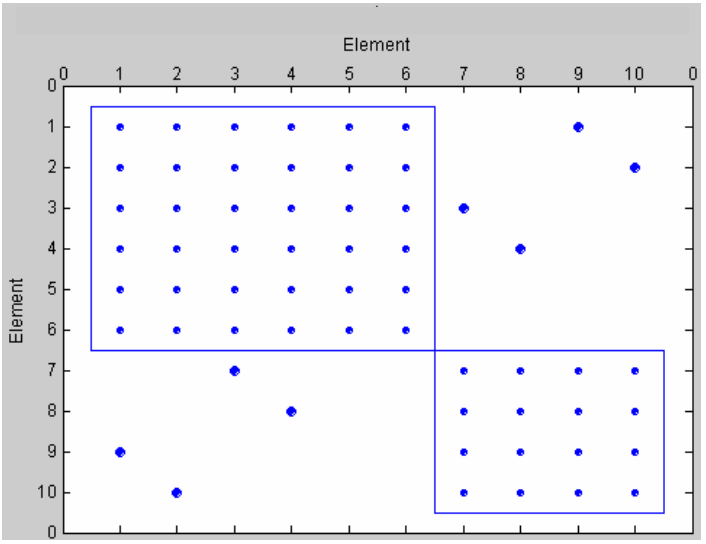


Figure III-17. Architecture obtenue avec notre algorithme

L'algorithme initial nous a fourni l'architecture représentée dans la figure III-18. Les deux modules identifiés ici diffèrent de ceux de l'architecture précédente. L'algorithme initial tend à construire deux modules de manière à rendre internes à ces modules les interactions à forte intensité (Figure III-16). Ainsi, en pénalisant les interactions externes plus que les interactions internes, l'algorithme initial ne peut qu'inclure les interactions externes qui sont fortes dans des modules pour baisser le coût de couplage total. Nous remarquons alors que les interactions externes finales dans la figure III-18 sont plus nombreuses mais avec une plus faible intensité et que les modules sont moins denses.

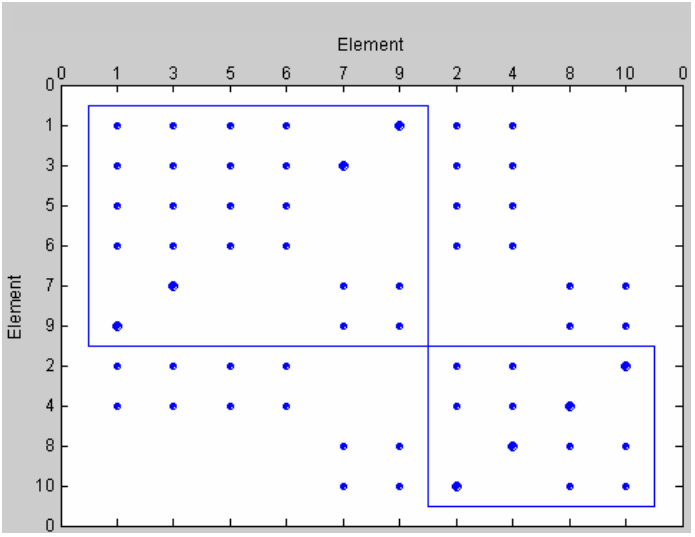


Figure III-18. Architecture obtenue avec l'algorithme initial

Ce dernier exemple illustre l'importance de l'utilisation des indicateurs MSIi' et MSIe' dans le calcul des coûts de couplages internes et externes aux modules. Ils permettent en effet de réduire l'influence de la relativité des valeurs internes et externes en privilégiant la création de modules dont la densité prime sur la taille.

3.6.3. Test sur une architecture hybride

La DSM représentée dans la figure III-19 montre l'exemple d'une architecture attendue dans laquelle il y a des modules couplés et des éléments intégrateurs.

	1	2	3	4	5	6	7	8	9	10	11	12
1		4	5	6			9				3	
2	4		0	8				3				4
3	5	0		7								6
4	6	8	7		5							8
5				5		3	8				7	
6					3		7		5			3
7	9				8	7				6		
8		3							8	9	6	
9						5		8		6		5
10							6	9	6			
11	3				7			6				7
12		4	6	8		3			5		7	

Figure III-19. DSM hybride de dimension 12 et l'architecture attendue

L'algorithme initial propose l'architecture représentée dans la figure III-20. On remarque que les deux éléments intégrateurs 11 et 12 ont été intégrés respectivement dans le premier et deuxième module.

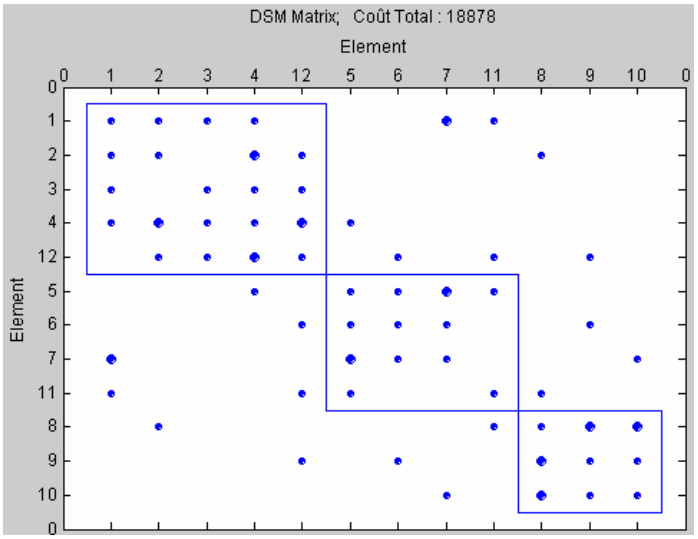


Figure III-20. Architecture obtenue avec l'algorithme initial

En utilisant notre algorithme, et en fixant IC= 0.5 et en identifiant directement l'élément 11 comme intégrateur, nous obtenons l'architecture représentée dans la figure III-21. On remarque que cette architecture est la même que celle qui était attendue. Il est intéressant de constater que le caractère intégrateur de l'élément 11 vient du fait qu'il couple les 3 autres modules identifiés et l'élément intégrateur 12.

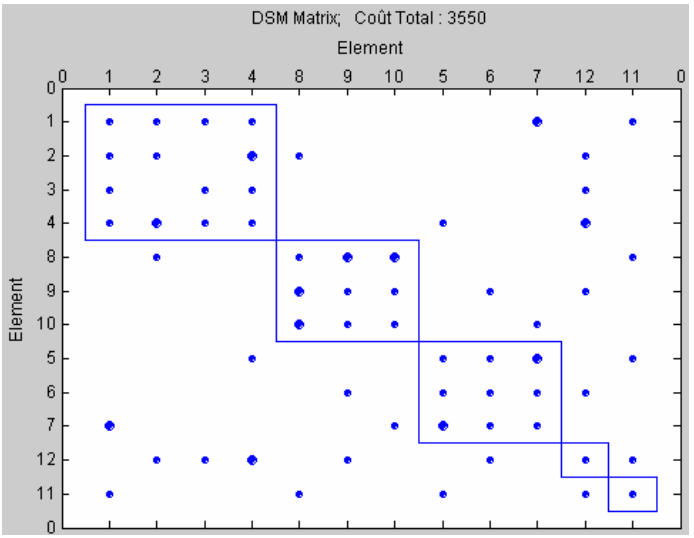


Figure III-21. Architecture obtenue avec notre algorithme

Il apparaît clairement qu'avec l'utilisation du seuil IC et une éventuelle aide manuelle, on arrive à identifier les éléments intégrateurs. Rappelons pour finir que l'existence de ces éléments va à l'encontre du principe même du clustering qui vise à minimiser les interactions externes et maximiser les interactions internes. Ce principe, appliqué sur les éléments intégrateurs, fait qu'il existe toujours un module qui peut accueillir ces éléments et réduire de ce fait le coût de couplage total.

4. Synthèse

Dans ce chapitre, nous avons présenté l'algorithme de clustering que nous utiliserons dans la suite de ce travail pour identifier l'architecture sous-jacente dans une DSM.

En vue d'obtenir un algorithme « relativement »[17] satisfaisant, nous sommes partis d'un algorithme hiérarchique du type bottom-up utilisant une fonction de pénalisation comme fonction objectif et faisant appel au recuit simulé pour explorer le plus grand nombre de solutions possibles.

Nous avons introduit deux coefficients correcteurs pour les fonctions coûts, ces coefficients sont issus de l'indicateur MSI développé par Whitfield et al. [2002]. Ceci nous a permis d'optimiser les capacités de l'algorithme à identifier les modules sous la forme d'éléments interagissant fortement (en nombre et en intensité) entre eux et faiblement avec les éléments appartenant aux autres modules.

Pour faciliter l'identification des éléments intégrateurs, nous avons automatisé l'identification des éléments comme intégrateurs en adoptant un seuil de couplage (IC) et nous avons permis à l'utilisateur d'identifier en amont les éléments qu'il perçoit comme étant intégrateurs.

[17] Par rapport à l'utilisation que nous faisons d'un algorithme de clustering et en comparaison avec les algorithmes de Idicula et de Thebeau

Enfin, la comparaison à l'algorithme initial montre que la limitation du nombre de paramètres facilite le réglage de l'algorithme et réduit le temps d'affinage des résultats.

L'algorithme final que nous avons obtenu a montré son efficacité que ce soit en fréquence ou en pertinence dans l'obtention d'architectures satisfaisantes et surtout identiques à ce que l'utilisateur peut attendre.

CHAPITRE IV

CHAPITRE IV

UNE METHODE POUR LE DEVELOPPEMENT DES ARCHITECTURES

Dans le cadre de l'IS, nous nous intéressons à la conception modulaire et plus précisément au principe de modularité. La conception d'un produit complexe peut être facilitée par la conception appropriée de son architecture. L'IS nous permet d'élargir l'objectif de pilotage du projet de conception, du produit vers le projet dans sa globalité. Ainsi, la conception du produit nécessite une structuration et une conduite efficace du projet. De la même manière que l'architecture du produit est un préalable à une conception détaillée satisfaisante, l'architecture de l'organisation du projet est un préalable à une conduite efficace du projet.

Nous distinguons deux types de projet de conception, les projets totalement innovants et les projets de reconception :

- L'organisation d'un projet totalement innovant se construit parallèlement à la définition du produit. Ces projets nécessitent des itérations successives pour corriger et affiner les résultats de l'activité de conception mais aussi pour adapter les plans d'action des équipes du projet. La nécessité de ces itérations est liée à la levée des incertitudes et à l'apport de connaissances, c'est-à-dire, à la qualité et à la quantité des données disponibles. Ces données sont progressivement enrichies et affinées, au cours du projet, d'une part, en approfondissant la spécification du produit et de ses constituants (espace des concepts, selon la théorie C-K [Hatchuel et Weil, 2002]), et d'autre part, en explorant ou en réutilisant des éléments de solution (espace des connaissances).
- Les projets de reconception correspondent à la reconduite de solutions développées dans des projets antérieurs. Ils portent sur des produits similaires tout en comportant une "petite" part de nouveauté. Des modifications peuvent être apportées sur le domaine du produit, des processus ou des acteurs. D'un point de vue structurel, si des modifications affectent l'un des domaines structurant le projet alors ces modifications doivent être propagées pour identifier les modifications dans l'architecture du projet.

Dans le cadre de ces deux situations de conception, nous pensons que la caractérisation des architectures des domaines du projet doit permettre d'améliorer les performances de l'activité de conception. En effet, pour les projets de conception des produits complexes, l'un des concepts clés que propose l'IS est de stratifier le processus de conception en se basant sur la décomposition du système. Cependant le passage d'un niveau de décomposition à un autre peut être rapide surtout quand la politique de conception de l'entreprise n'est pas basée sur la conception modulaire.

Considérons le sous-système GMP (Groupe MotoPropulseur) d'une automobile et sa décomposition physique (FigureIV-1) telle qu'elle est définie chez le constructeur automobile avec qui nous avons collaboré, on remarque qu'au premier niveau il y a un seul système, au deuxième deux sous-systèmes et au troisième niveau, 24 constituants.

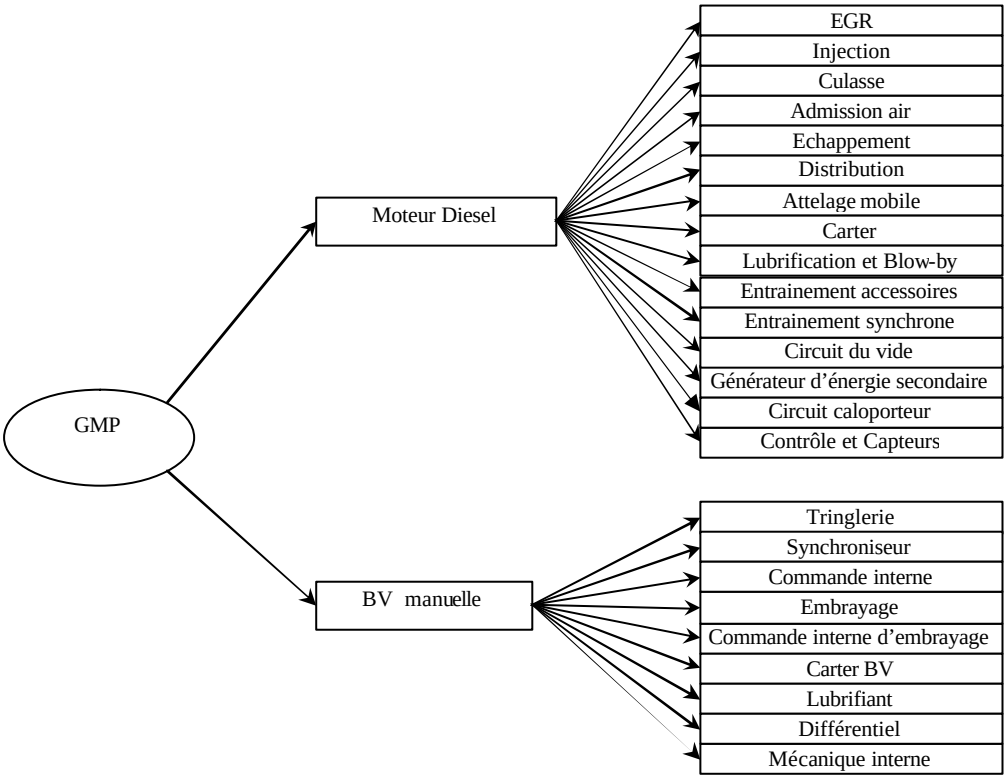


Figure IV-1. Décomposition physique du GMP

Ainsi, même en appliquant l'IS et la décomposition progressive qu'elle propose pour faire face à la grande complexité des produits du secteur automobile, on est confronté comme le montre la figure IV-1 à l'explosion, d'une strate à l'autre, du nombre d'éléments à concevoir. La difficulté n'est pas liée à l'allocation des ressources nécessaires mais à la propagation des contraintes et à la négociation des spécifications entre les éléments et entre les deux strates successives.

L'outil de conception conjointe des architectures des domaines du produit que nous présentons dans ce chapitre a pour but de s'interfacer entre les deux derniers rangs de la

décomposition présentée ci-dessus pour proposer une étape intermédiaire dans la décomposition des systèmes (figure IV-2). En se basant sur la typologie d'architecture d'Ulrich (1995) et sur la définition des modules, l'outil permet à l'architecte système de simuler les architectures des domaines du produit et, si elles sont validées, les utiliser pour :

- spécifier les interfaces fonctionnelles ou physiques (figer les interfaces externes aux modules avant de figer les interfaces internes entre les éléments appartenant à un même module).
- optimiser les processus d'ingénierie et d'intégration en utilisant les architectures validées comme étapes permettant le passage d'une strate à une autre et ainsi maîtriser la complexité des tâches à accomplir.

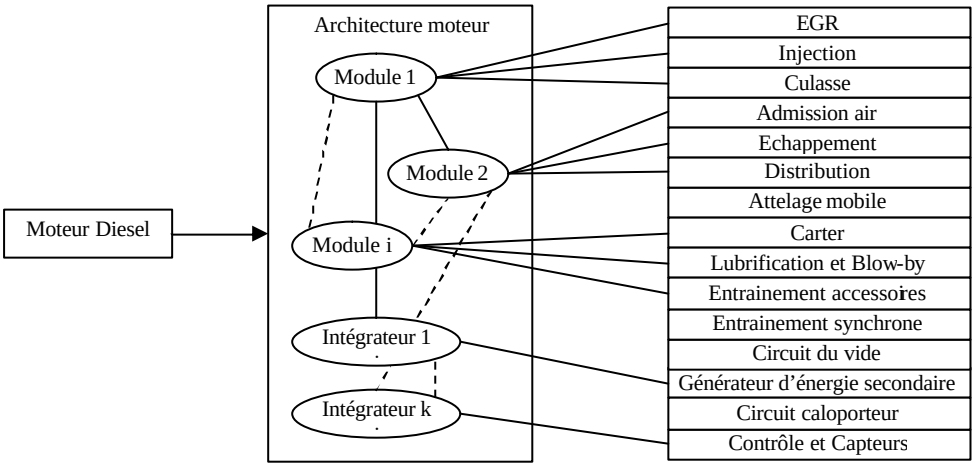


Figure IV-2. Positionnement de la méthode d'architecture sur le moteur

Dans la suite de ce chapitre, nous présentons d'abord les quatre situations d'utilisation possible de l'outil aidant à la construction des architectures. Cette typologie des situations repose sur le choix des outils matriciels de modélisation qu'on a adoptés à savoir les matrices d'incidence et les DSM. Nous proposons une démarche pour la construction des données utilisées en entrée de la méthode. Puis, nous expliquons une à une les méthodes proposées à chacune des situations avec quand c'est nécessaire l'utilisation d'un traitement flou.

1. Construction des architectures : quatre situations d'utilisation possibles

1.1. Cadre général

L'architecture d'un système se détermine à travers l'architecture des sous-systèmes qui le composent. L'algorithme de clustering que nous avons présenté dans le paragraphe précédent est l'outil que nous allons utiliser pour révéler l'architecture des systèmes. Etant donné que

l'algorithme utilise en entrée des DSM, nous devons construire les DSM de chacun des systèmes dont nous voulons étudier l'architecture.

La question qui se pose à nous provient en partie des difficultés qu'on a rencontrées sur le terrain pour construire les DSM: Que faire quand les DSM ne sont pas disponibles ou difficiles à construire ?

Une partie de la réponse à cette question provient de l'existence de deux types de méthodes matricielles pour la modélisation des couplages dans un système : les matrices intra-domaines à savoir les DSM et les matrices inter-domaines que nous avons décidé d'appeler Matrices d'Incidence (MI). Ainsi dans le cas où les DSM ne sont pas accessibles, est-il possible d'utiliser les MI ?

Dans la suite de ce chapitre, nous allons démontrer que :

- Premièrement : les MI ou plutôt l'information qu'elles recèlent est un élément clé du processus de conception et le fondement même de la compétence des architectes systèmes. Il s'ensuit que l'information nécessaire à la construction de ces MI est disponible relativement tôt dans le processus de conception ;
- Deuxièmement : à travers un traitement flou qui sera approfondi ultérieurement, nous montrons qu'une seule MI permet d'obtenir deux DSM. Il s'ensuit que par gain de temps et pour réduire la sollicitation de nos interlocuteurs, nous préférons construire des MI ;
- Enfin : dans les différentes situations de construction des architectures des domaines que nous présenterons et en tenant compte de la relation hiérarchique entre les MI et les DSM, nous démontrerons que les MI interviennent dans la majorité des situations et qu'elles permettent de créer de la redondance dans les résultats, ce qui permettra de les affiner.

L'analyse de l'activité de l'architecte système nous a permis d'identifier l'existence de quatre situations élémentaires de construction des architectures des systèmes. Ces situations peuvent être combinées pour décrire une situation réelle : par exemple l'architecture d'un projet de conception ou l'architecture d'un produit.

Considérons un système composé de trois domaines A, B et C modélisés par des DSM liées par deux MI. Ce système représenté dans la figure IV-3 est un système générique avec lequel on peut expliquer tout système ayant un degré de complexité plus grand –en recourant à la décomposition hiérarchique des systèmes.

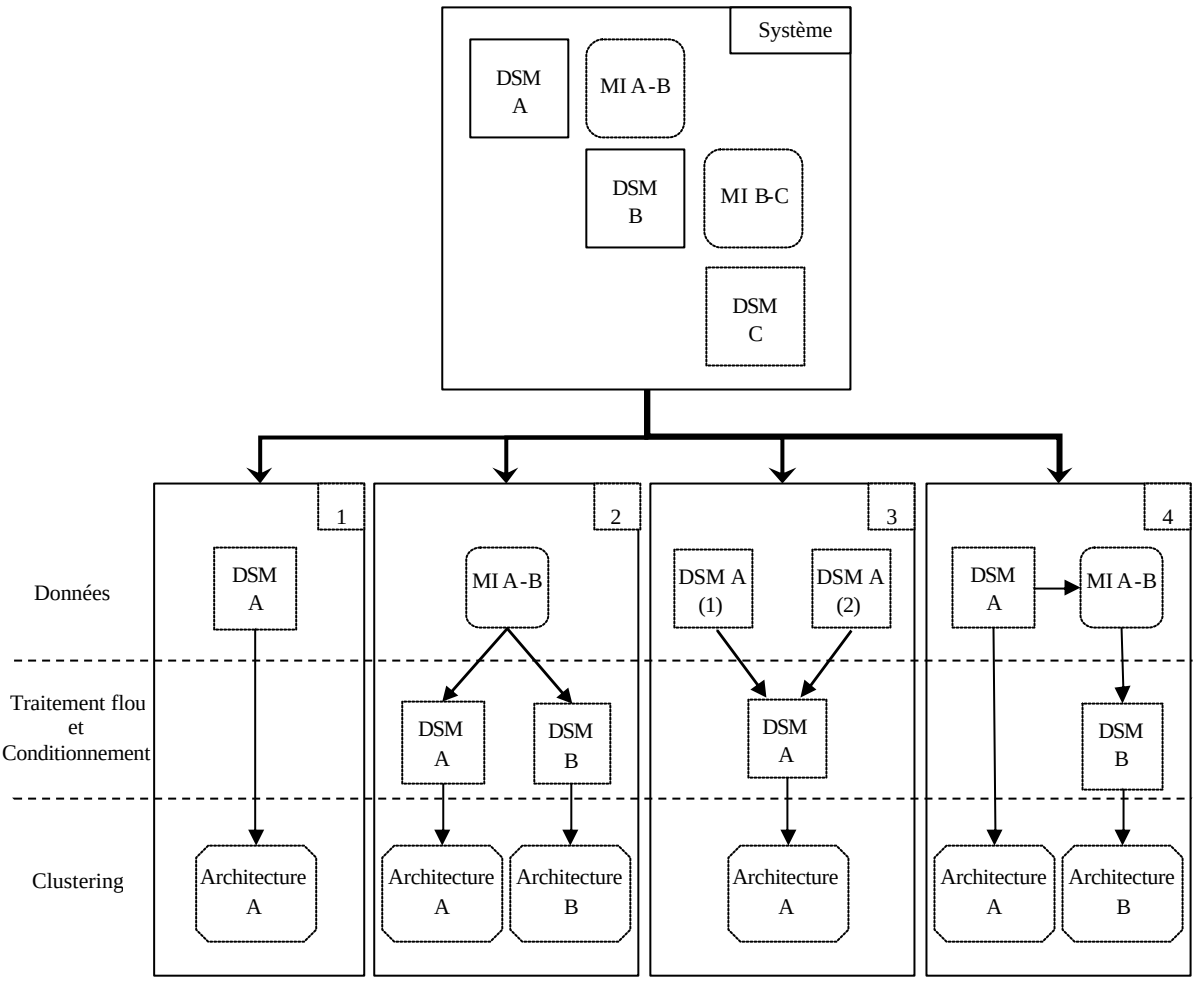


Figure IV-3. Les quatre situations de construction des architectures

Les quatre situations que nous avons identifiées sont déterminées à travers le type de données disponibles. Pour chaque situation et donc pour chaque type de données disponibles, nous proposons une méthode de construction des architectures.

1.2. Les quatre situations élémentaires de construction des architectures

Nous allons énumérer dans ce qui suit les différentes situations en présentant leurs caractéristiques et les méthodes utilisées pour construire les architectures :

- Première situation : elle est caractérisée par la disponibilité d'une DSM par domaine. Si cette DSM n'est pas le résultat d'un traitement antérieur, elle est alors construite manuellement par les acteurs du projet. Dans ce cas, nous présenterons des règles de construction pour les méthodes matricielles que nous appliquerons sur l'exemple d'une DSM Fonctions Systèmes.
- Deuxième situation : elle est caractérisée par la donnée d'une MI (ou plusieurs n'ayant aucun domaine en commun). Dans ce cas, la matrice d'incidence est nécessairement

construite manuellement, nous proposons alors des règles pour la construction de ces MI. Ensuite, nous proposons un traitement flou pour construire les DSM A et DSM B en partant de MI A-B et des opérations de conditionnement pour transformer les DSM simulées afin de les préparer pour le clustering.

- Troisième situation : elle est caractérisée par la redondance d'informations concernant un domaine donné. Pour un domaine A, nous pouvons avoir deux DSM A qui peuvent soit être construites par deux acteurs différents, soit être issues de deux MI différentes (situation 2), soit un mélange des deux possibilités précédentes. Nous proposons alors une méthode pour préparer les DSM données quand c'est nécessaire et une autre pour les agréger et générer la DSM résultante pour le clustering
- Quatrième situation : elle est caractérisée par la disponibilité d'une DSM A et d'une MI A-B. Nous proposons d'abord un conditionnement de la DSM A, ensuite un traitement flou qui utilise ces deux matrices en entrée pour simuler une DSM du domaine B (DSM B) qui sera utilisée par l'algorithme de clustering pour proposer une architecture du domaine B.

2. Collecte et structuration des données

Avant de développer la méthode pour chacune des situations élémentaires, nous proposons aux utilisateurs des règles et des conseils pour la construction manuelle d'une matrice qu'elle soit une DSM ou une MI.

2.1. Construction directe d'une matrice d'incidence

Dans ce travail, nous conférons aux matrices d'incidence un rôle central dans l'architecture global d'un projet de conception. Le choix des matrices d'incidence comme donnée principale pour la simulation des architectures des domaines du projet est dictée par deux constations :

- Même si les matrices d'incidence numériques ne sont pas utilisées comme outil de modélisation, l'information nécessaire à leur construction, à savoir les couplages inter-domaines, sont souvent disponibles sous plusieurs formes : graphes, schémas bulle ou matrices non numériques ;
- L'information nécessaire à la construction des matrices d'incidence peut être disponible très tôt dans le processus de conception, spécifiquement dans le cadre des projets complexes où dans les phases préliminaires, une première identification de la composition du projet et de la structure globale du projet est réalisée.

Ainsi, le fait que l'information concernant le couplage entre les domaines du projet peut être disponible sous plusieurs formes et assez tôt dans le projet nous a permis de choisir les

matrices d'incidence comme point de départ pour la construction des architectures des domaines du projet.

Dans ce qui suit, nous allons présenter l'importance des matrices d'incidence. Ensuite nous proposerons un guide pour la construction de ces matrices, guide qui nous a servi pour assister les acteurs du projet dans leur construction.

2.1.1. Métier d'architecte système et rôle clé des matrices d'incidence

Dans le cadre d'un projet de recherche que nous avons mené avec un constructeur automobile français, nous avons pu analyser l'activité des architectes systèmes sur deux projets de conception l'un portant sur un moteur diesel, et l'autre sur une boîte de vitesse robotisée. Conformément à l'IS, les architectes manipulent des vues externes et internes du produit.

La vue externe du produit est représentée par l'espace des exigences (appelé aussi espace des prestations). La vue interne du produit, quant à elle, se compose des deux domaines que nous avons identifiés dans le premier chapitre, à savoir le domaine des Fonctions systèmes (vue fonctionnelle interne) et le domaine des composants (vue organique).

Les exigences sont des contraintes que le produit doit satisfaire. Ces exigences sont satisfaites à travers la réalisation des fonctions du produit. Ces fonctions quant à elles sont portées par les constituants du produit.

Nous considérons que les exigences sont des données du projet pour l'architecte système. Face à ces données, l'architecte doit garantir tout au long du projet la cohérence d'un côté entre la strate fonctionnelle et la strate organique du produit (face aux aléas et aux modifications) et d'un autre côté la cohérence de ces deux strates avec les exigences du produit.

En travaillant sur les architectures des domaines du produit, l'architecte système peut d'un côté piloter la cohérence de ces architectures et de ce fait garantir la propagation des contraintes et des modifications et d'un autre côté utiliser les architectures pour figer certains paramètres du produit tels que les modules fonctionnels ou organiques et leurs interfaces.

Dans l'entreprise, les architectes systèmes sont les garants du passage de la strate fonctionnelle à la strate organique. C'est pourquoi ils ont montré un grand intérêt pour un outil qui permet de simuler les architectures des domaines du produit et de formaliser les représentations des couplages et des architectures par des MI et des DSM.

2.1.2. Règles pour la construction de la matrice d'incidence

Dans ce paragraphe nous proposons un guide pour la construction des matrices d'incidence. Nous avons utilisé, affiné et validé ce guide lors de nos entretiens avec les ingénieurs de PSA.

L'écriture de ce guide est une nécessité pour les raisons suivantes :

- Même si les matrices d'incidence sont déjà utilisées pour la projection des spécifications, la façon de les remplir avec des valeurs numériques n'est pas usuelle pour les ingénieurs ;
- Les matrices d'incidence sont l'unique entrée sur laquelle se base la méthode que nous proposons dans la partie 4 (situation 2). De ce fait, leur consistance et leur cohérence doivent permettre d'assurer la qualité des résultats de notre méthode ;
- Les matrices d'incidence sont des matrices remplies par les acteurs du projet et non le résultat d'une quelconque opération logique ou automatique. De ce fait, le guide sert à uniformiser et harmoniser cette étape de construction pour réduire et prendre en compte des biais provenant de l'estimation humaine.

Le guide est composé de 5 étapes :

- créer la liste des éléments, appartenant à chacun des domaines, qui seront liés par la matrice d'incidence. Selon l'état d'avancement du projet, le niveau de décomposition utilisé dans les listes peut être plus moins fin.
- utiliser le même niveau de décomposition avec une granularité semblable dans les deux domaines comme représenté dans la figure IV-4 montrant la décomposition de deux produits A et B.

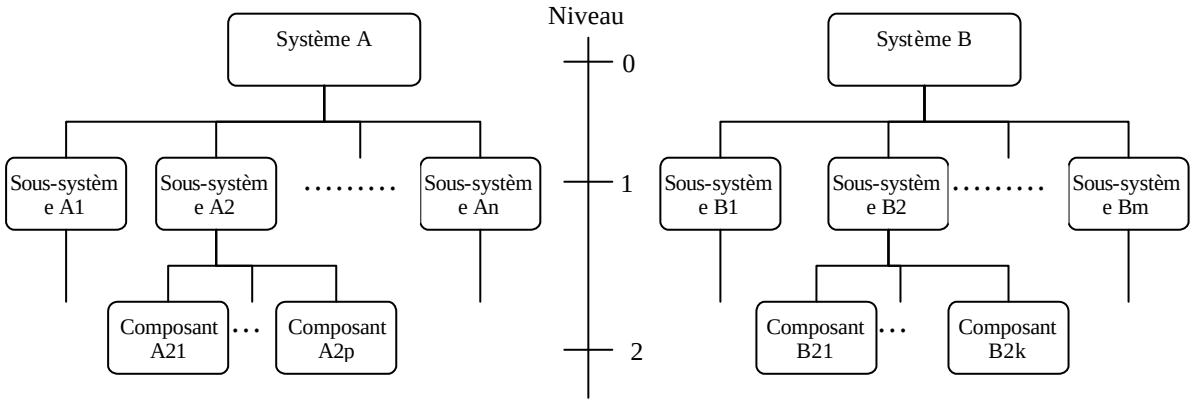


Figure IV-4. Coordination des niveaux de décomposition

- construire une matrice d'incidence avec en ligne la liste des éléments d'un domaine et en colonne la liste de l'autre domaine.
- adopter une caractérisation sémantique bidirectionnelle pour qualifier l'interaction entre deux éléments appartenant à deux domaines différents. Par exemple : il y a interaction lorsqu'on propage des contraintes dans un sens ou dans l'autre ; ou il y a interaction lorsqu'il est nécessaire d'arbitrer ou de négocier entre deux éléments.

- identifier les paires d'éléments qui interagissent pour renseigner la matrice d'incidence, d'abord de façon binaire (0/1). Nous rappelons à cet effet que seules les interactions directes doivent être identifiées. Dans le cas contraire, et selon la complexité du projet, nous allons obtenir une grande densité dans la matrice d'incidence. Il faut savoir alors, et nous le présenterons ultérieurement, que plus la matrice d'incidence est dense[18], moins les résultats de notre méthode sont pertinents.
- construire, à partir de la matrice binaire, une matrice d'incidence numérique en donnant une évaluation aux interactions. Nous considérons alors les hypothèses suivantes :
    - la valeur d'une interaction est un entier appartenant à]0, 10]
    - la valeur d'une interaction évalue l'intensité du couplage entre les éléments. Si le couplage est fort et que la dépendance est forte alors la valeur est proche de 10, dans le cas contraire elle est proche de 0.
    - Il est possible de considérer les éléments comme un ensemble d'attributs et d'évaluer ainsi la densité des couplages entre les attributs d'un élément en ligne et un autre en colonne.
    - Nous ne proposons pas de contraintes sur le nombre d'interactions fortes ou faibles, mais nous rappelons qu'un élément intervenant avec un grand nombre d'interactions ou avec des interactions toujours fortes influencera plus fortement la solution proposée par notre méthode. Alors si la position de l'élément n'est pas aussi importante, il faut limiter le nombre d'interactions ou diminuer le poids de ces interactions.

La matrice d'incidence ainsi construite est prête à être utilisée comme donnée d'entrée pour la construction des architectures des domaines du projet.

2.1.3. Exemple : matrice d'incidence FS-COMP

Dans le cadre de ce travail de thèse, nous avons eu la possibilité de collaborer avec un constructeur automobile leader mondial des moteurs Diesel. Ce constructeur est PSA Peugeot Citroën. Le projet de développement qui a joué le rôle de cadre applicatif à notre travail est un moteur Diesel en cours de développement.

L'information nécessaire pour la création et le remplissage des matrices d'incidence existe dans l'entreprise, que ce soit sous forme matricielle ou sous forme de graphes. Plus spécifiquement, dans le cadre d'une action de reconception, ces informations existent nécessairement à travers la modélisation antérieure du produit et du projet. Ceci nous amène à caractériser la situation de conception rencontrée chez PSA de la manière suivante :

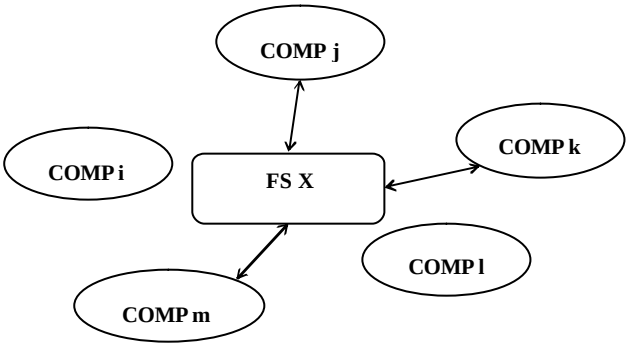
[18] La densité est le rapport du nombre d'interactions non nulles par le nombre d'interactions possibles

- Les domaines du projet ainsi que les éléments qui les composent sont clairement identifiés.
- Les concepteurs et plus précisément les architectes systèmes manipulent des graphes qui permettent de lier des éléments du projet entre eux, ces graphes peuvent être formalisés par des DSM binaires (O pour inexistence d'interactions et X pour le cas contraire).

Dans la pratique, la génération des matrices d'incidence se fait de deux manières qui peuvent être combinées :

- Par des interviews pour créer les matrices lorsque la modélisation n'existe pas.
- Par traduction et transformation des modèles déjà créés (sous forme de graphes ou de matrices).

La figure IV-5 montre un exemple de graphe utilisé chez PSA, appelé « schémas bulle » ainsi que sa traduction en matrice d'incidence.



	. . .	COMP i	COMP j	COMP k	COMP l	COMP m	. . .
. .							
FS X			X	X		X	
. .							

Figure IV-5. Traduction des schémas bulle en MI

La matrice d'incidence binaire construite à partir des graphes obtenus chez PSA est représentée dans la figure IV-6.

FS / Comp	Resp	Alim carb	Dépol	Comb	CPC	PLV	EFF	CES	ACV	IES	THE	COU	Vol Fonc
EGR	X	0	X	0	0	0	0	0	X	X	X	0	X
Injection Carburant	0	X	0	X	0	0	X	0	X	X	X	0	X
Culasse Assemblée	X	X	X	X	0	X	X	X	X	0	X	0	X
Admission d'air	X	X	X	X	0	0	X	X	X	X	X	0	X
Echappement	X	0	X	0	0	X	X	0	X	X	X	0	X
Distribution	X	X	0	0	0	X	0	X	X	0	0	0	X
Attelage mobile	X	0	0	X	X	X	X	0	X	0	X	X	X
Carter	0	0	0	X	X	X	X	0	X	0	X	X	X
Lubrification et Blow-by	X	0	0	0	X	X	0	X	X	X	X	0	X
Entraînement accessoires	0	0	0	0	X	0	X	X	X	0	X	0	X
Entraînement synchrone	0	0	0	0	X	X	0	X	X	0	X	0	X
Circuit vide	X	0	0	0	0	X	0	X	0	0	0	0	X
Circuit caloporteur	0	0	0	0	0	X	X	X	X	0	X	0	X
GES	0	0	0	0	0	X	X	X	X	0	X	0	X
CC	X	X	X	X	0	X	0	0	0	X	X	X	X

Figure IV-6. MI binaire FS-COMP

Avec :

Abréviation	Nom de la Fonction Système
Resp	Respiration
Alim carb	Alimentation Carburant
Dépol	Dépollution
Comb	Combustion
CPC	Conversion Pression Couple
PLV	Pertes par frottement, Lubrification et Ventilation
EFF	reprise d'EFFort
CES	Conversion et dérivation des Energies Secondaires
ACV	ACoustique Vibratoire
IES	Sensorique et contrôle
THE	Thermique
COU	Couplage
Vol Fonc	Volumes fonctionnels

Abréviation	Nom du composant
EGR	*Exhaust Gas Recycle*, système permettant la réinjection des gaz brulés dans l'admission
GES	Générateur et stockage d'énergie
	Injection Carburant
	Culasse Assemblée
	Admission d'air
	Echappement
	Distribution
	Attelage mobile
	Carter
	Lubrification et Blow-by
	Entraînement accessoires
	Entraînement synchrone
	Circuit vide
	Circuit caloporteur
CC	Capteurs et Commande

Tableau IV-1. Liste des FS et des composants avec leurs abréviations

Sur la base de cette matrice binaire et en se servant du guide de construction des matrices d'incidence, l'équipe de conception du moteur a construit la DSM numérique représentée dans la figure IV-7. Après discussion sur l'intensité de certaines valeurs, le choix définitif a été fait par l'architecte système.

FS / Comp	Resp	Alim carb	Dépol	Comb	CPC	PLV	EFF	CES	ACV	IES	THE	COU	Vol Fonc
EGR	8	0	9	0	0	0	0	0	6	8	7	0	6
Injection Carburant	0	8	0	8	0	0	7	0	7	8	5	0	8
Culasse Assemblée	5	5	6	5	0	8	7	5	7	0	8	0	8
Admission d'air	9	9	6	9	0	0	4	5	6	9	3	0	7
Echappement	8	0	9	0	0	6	5	0	6	6	8	0	6
Distribution	9	9	0	0	0	7	0	8	8	0	0	0	7
Attelage mobile	5	0	0	7	9	7	7	0	8	0	8	9	8
Carter	0	0	0	8	6	8	9	0	8	0	8	9	9
Lubrification et Blow-by	5	0	0	0	8	9	0	4	5	3	8	0	7
Entraînement accessoires	0	0	0	0	6	0	8	9	5	0	6	0	8
Entraînement synchrone	0	0	0	0	8	9	0	9	8	0	6	0	7
Circuit vide	8	0	0	0	0	8	0	9	0	0	0	0	8
Circuit caloporteur	0	0	0	0	0	5	8	5	6	0	9	0	7
GES	0	0	0	0	0	4	5	9	5	0	5	0	7
CC	7	9	7	8	0	6	0	0	0	9	9	8	4

Figure IV-7. MI FS-COMP numérique

Dans la suite de ce document, les autres matrices d'incidence seront considérées comme étant données, le processus de construction ayant été appliqué de la même manière.

2.2. Construction directe des DSM

A travers les scénarios de construction des architectures des domaines du projet, on remarque que la première et la quatrième situation sont caractérisées par la disponibilité d'une ou deux DSM comme donnée pour l'identification des architectures.

Ces DSM utilisées comme données sont des DSM qui ont été construites –quand c'est possible- avec la participation des acteurs du projet. Les règles de construction qui régissent le remplissage de ces DSM sont les mêmes que celles qui ont été présentées pour la construction des matrices d'incidences. L'unique différence réside dans le fait qu'on a la même liste d'éléments en ligne et en colonne.

2.2.1. Exemple : DSM FS

La DSM FS numérique, construite manuellement, est présentée dans la figure IV-8. Cette DSM est relativement dense : 73% (la densité est le rapport entre le nombre d'interactions et le nombre d'interactions possibles), mais cette DSM ne subira aucune modification.

	Resp	Alim carb	Dépol	Comb	CPC	PLV	EFF	CES	ACV	IES	THE	COU	Vol Fonc
RESP	1	9	7	8			6	5	6	8	7		6
ALIM CARB	9	2	9	9			7		7	8	5		7
DEPOL	7	9	3	9		8	8		7	7	9		7
COMB	8	9	9	4	8	4			8	9	8		4
CPC				8	5	9	9	8	8		7	8	8
PLV			8	4	9	6	7	8	5	6	8		6
EFF	6	7	8		9	7	7	8	8		8	9	8
CES	5				8	8	8	8	3		5		6
ACV	6	7	7	8	8	5	8	3	9	5		7	
IES	8	8	7	9		6			5	10	5		5
THE	7	5	9	8	7	8	8	5		5	11	8	
COU					8		9		7		8	12	8
VOL FONC	6	7	7	4	8	6	8	6		5		8	13

Figure IV-8. DSM FS construite par les acteurs du projet

Cette DSM sera utilisée dans les situations suivantes :

- dans la première situation, pour analyser l'architecture de la DSM construite directement par les acteurs du projet,
- dans la deuxième situation, pour analyser le résultat de la méthode proposée,
- dans la quatrième situation, pour propager les contraintes d'architecture d'un domaine vers un autre.

3. Première situation : Identification d'une architecture à partir d'une DSM donnée

Cette situation est décrite sur la figure IV-9. Elle correspond à une DSM construite directement par les acteurs du projet.

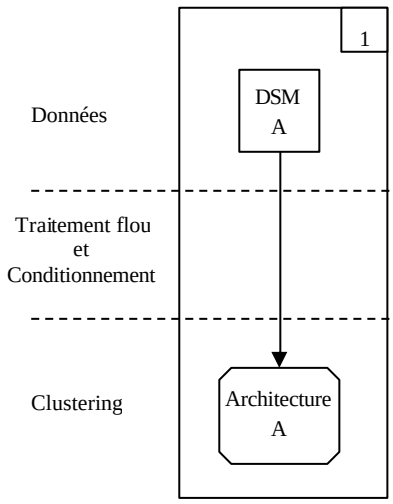
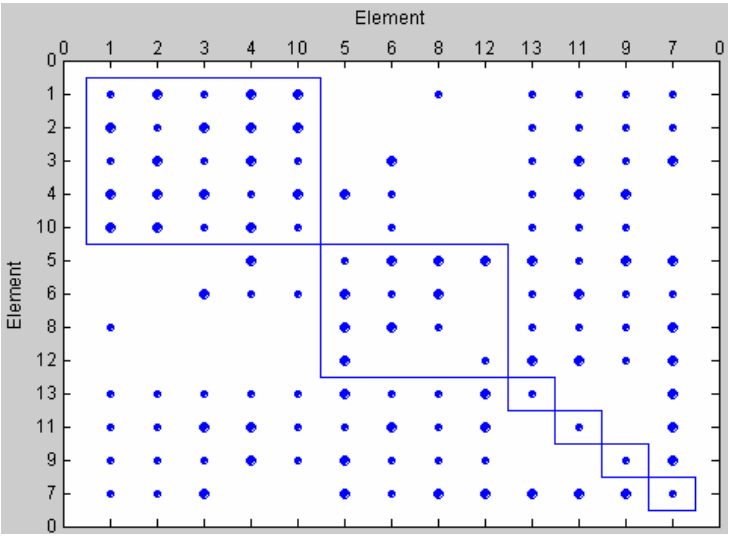


Figure IV-9. Schéma de la première situation

Pour illustrer cette situation, nous disposons de la DSM FS présentée précédemment. Avec un IC=0.8, nous obtenons l'architecture représentée dans la figure IV-10.



1	2	3	4	5	6	7	8	9	10	11	12	13
RESP	ALIM CARB	DEPOL	COMB	CPC	PLV	EFF	CES	ACV	IES	THE	COU	VOL FONC

Figure IV-10. Architecture de la DSM FS

Cette architecture nous suggère les remarques suivantes :

- L'algorithme de clustering a identifié deux modules et quatre fonctions intégratrices ;
- Le premier module est composé des FS : Respiration, Alimentation carburant, Dépollution, Combustion et Sensorique. Les quatre premières fonctions forment la phase de précombustion et de combustion du moteur. A coté de ces fonctions nous retrouvons la sensorique, ceci nous renseigne sur la présence indispensable du contrôle dans cette phase du fonctionnement du moteur.
- Le deuxième module est composé des FS : Conversion pression couple, PLV, Conversion énergies secondaires et Couplage. Ces fonctions se situent toutes dans la phase de postcombustion.
- Les FS intégratrices identifiées sont : Reprise d'effort, Acoustique vibratoire, Thermique et Volumes fonctionnels. Parmi ces fonctions, les acteurs confirment le caractère intégral des trois dernières fonctions puisqu'elles sont connues pour impacter la plupart des autres fonctions. Le caractère intégrateur de la reprise d'effort s'explique à travers l'importance de cette fonction dans la réalisation de la cohésion du moteur, cohésion qui s'explique si on considère que le carter, l'attelage mobile et la culasse sont les composants porteurs de cette fonction dans l'architecture organique.

4. Deuxième situation : Construction de deux DSM à partir d'une MI

Dans cette partie, nous présentons la méthode utilisée pour générer les DSM de deux domaines à partir de la matrice d'incidence numérique (figure IV-11). La méthode adoptée se base principalement sur un traitement flou des données contenues dans les matrices d'incidence [Harmel et al., 2007] pour générer deux matrices DSM.

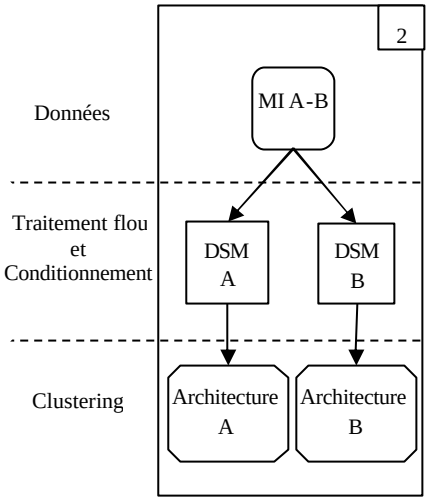


Figure IV-11. Schéma de la deuxième situation

4.1. Présentation globale du processus de génération des DSM

Le processus permettant de passer d'une matrice d'incidence numérique liant les éléments de deux domaines différents à deux DSM représentant les interactions internes à chaque domaine est basé sur deux étapes principales. La première régit et explique le passage d'une matrice d'incidence à deux DSM, elle prend la forme de règles. La deuxième représente la mise en œuvre des principes énoncés dans les règles et prend la forme d'un traitement flou.

4.2. Les règles et leurs applications

4.2.1. Introduction : les graphes

Considérons le graphe présenté dans la figure IV-12. Ce graphe a la particularité de représenter des liens (des arêtes) entre deux types de sommets (ou deux espaces), les A et les B. Une arête lie un élément de A à un élément de B (on ne peut pas trouver deux éléments de A ou de B liés par une arête). Lorsque l'arête existe, elle a un poids non nul, appartenant à l'intervalle]0, 10].

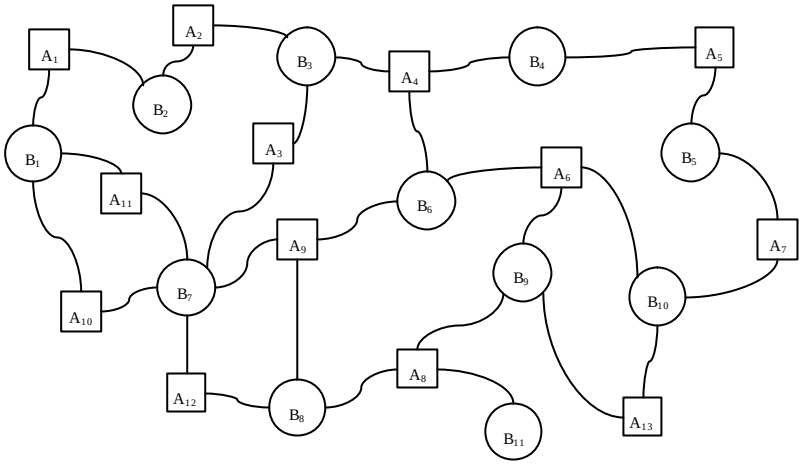


Figure IV-12. Exemple de graphe avec deux domaines

Ce graphe peut être représenté d'une manière différente, sous la forme d'une matrice d'incidence. On obtient alors la matrice d'incidence présentée dans la figure IV-13.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**	**A_6**	**A_7**	**A_8**	**A_9**	**A_{10}**	**A_{11}**	**A_{12}**	**A_{13}**
B_1	X									X	X		
B_2	X	X											
B_3		X	X	X									
B_4				X	X								
B_5					X		X						
B_6				X		X			X				
B_7			X						X	X	X	X	
B_8								X	X			X	
B_9						X		X					X
B_{10}						X	X						X
B_{11}								X					

Figure IV-13. Traduction du graphe en MI

Notre objectif dans cette partie est de proposer une méthode qui permet de représenter les deux sous-graphes des A_i et des B_i en partant du graphe initial. En utilisant les outils matriciels que nous avons présentés dans le chapitre II, cet objectif devient : proposer une méthode qui permet de construire la DSM des A_i et la DSM des B_i en partant de la matrice d'incidence présenté en figure IV-13.

Dans la théorie des graphes, la notion de chemin est importante. Nous allons l'utiliser pour introduire la méthode que nous proposons.

Nous avons déjà muni les arêtes de poids, nous savons que la théorie des graphes permet de réaliser le passage présenté dans la figure IV-14. Cependant, la question est comment exprimer la valeur du lien direct entre A_i et A_j. L'intensité des interactions n'étant pas une distance, nous ne pouvons pas utiliser la somme pour estimer l'intensité l'interaction entre A_i et A_j.

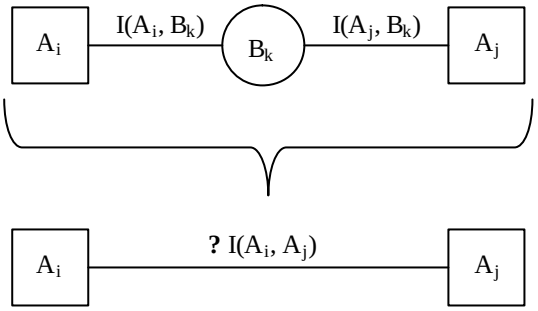


Figure IV-14. Modélisation de la problématique de distance

En réponse à cette dernière remarque, nous introduisons des règles qui nous permettent d'évaluer les interactions entre les éléments d'un même domaine en partant d'une matrice d'incidence.

4.2.2. Enoncé général des règles

Dans ce qui suit, nous allons utiliser la terminologie propre aux DSM ainsi que la notation précédente, pour les formuler :

Règle 1. Si A_i et A_j interagissent avec B_k, alors A_i et A_j sont couplés.

Règle 2. L'intensité du couplage entre A_i et A_j dépend de l'intensité des couplages entre A_i et B_k et A_j et B_k.

Règle 3. Les interactions entre les éléments de B se construisent symétriquement à ceux des A

La mise en œuvre de ces règles nécessite l'explicitation de la **Règle 2**. Auparavant, nous allons mettre en situation nos règles en les appliquant à l'architecture du produit.

4.2.3. Formulation des règles sur un exemple

Pour expliciter le principe de ces règles, considérons par exemple le cas d'une MI Composants-Fonctions Systèmes à partir de laquelle nous souhaitons obtenir la DSM Composants et la DSM Fonctions Systèmes.

Nous présentons sur la base de cet exemple la formulation des règles adoptées :

Règle 1. Si deux Fonctions Systèmes FS1 et FS2 interagissent avec un composant C alors FS1 et FS2 sont couplées à travers ce composant.

Règle 2. L'intensité du couplage entre FS1 et FS2 est liée à l'intensité de leur couplage avec le composant C.

Règle 3. Le domaine des composants peut être construit symétriquement aux fonctions systèmes.

Dans ce qui suit, nous allons essayer d'expliquer les fondements de ces règles en nous référant à l'exemple proposé.

Considérons un composant caractérisé par un ensemble d'attributs ou de paramètres [X_1, X_2, …, X_n], on distingue alors deux cas :

- Cas 1 : FS1 et FS2 impactent en commun un sous-ensemble non vide des paramètres caractérisant C. Dans ce cas, le processus de conception nécessite pour spécifier le composant de C de fixer les paramètres qui le caractérisent et donc de propager les contraintes de FS1 et de FS2 sur ces paramètres, il s'en suit clairement que FS1 et FS2 sont couplés.
- Cas 2 : FS1 et FS2 ne contraignent pas les mêmes paramètres de C, cela ne doit pas nous faire oublier que les paramètres caractérisant un composant sont rarement indépendants. Nécessairement alors, FS1 et FS2 sont couplées à travers le couplage entre les paramètres caractérisant le composant C.

Concernant la règle 2, nous rappelons qu'elle affirme seulement que les intensités sont liées et qu'elle ne précise pas comment. Il est nécessaire pour lier les intensités de se munir d'un formalisme adapté. Dans ce travail, nous avons opté pour l'utilisation d'un traitement flou.

4.3. Le processus flou

Les matrices d'incidence sont toutes construites par les acteurs du projet selon leur perception et interprétation des données disponibles dans l'entreprise.

Par ailleurs, le principal objectif de la logique floue est de pouvoir modéliser, imiter et simuler les fonctionnalités du raisonnement humain dans des situations incertaines ou imprécises. La logique floue permet ainsi de manipuler des données qualitatives plutôt que quantitatives [Zadeh, 1975].

Il apparaît ainsi judicieux de proposer un traitement flou pour construire les architectures des domaines du projet en partant des matrices d'incidence. Cette méthode permet de limiter l'effet des imprécisions sur les estimations fournies par les acteurs du projet pour caractériser les interactions inter-domaines. On retrouve le même type de traitement flou dans la modélisation des systèmes experts [Graham, 1991; Kandel, 1992].

Le processus flou doit permettre de mettre en œuvre toutes les règles de construction permettant le passage d'une matrice d'incidence à deux DSM. Afin d'expliciter le processus flou, nous reprenons l'exemple de la matrice d'incidence FS-COMP (figure IV-7).

Selon la troisième règle, les deux domaines sont construits d'une façon symétrique. De ce fait, nous allons expliquer la construction de la DSM FS, la DSM COMP sera construite avec le même processus.

Le processus flou permet de calculer la valeur de l'interaction entre deux fonctions F_j et F_k qui interagissent avec un composant C_i. Le système flou a la structure présentée dans la figure

IV-15. Le traitement flou que nous proposons permet de générer une DSM FS pour chaque composant C_i.

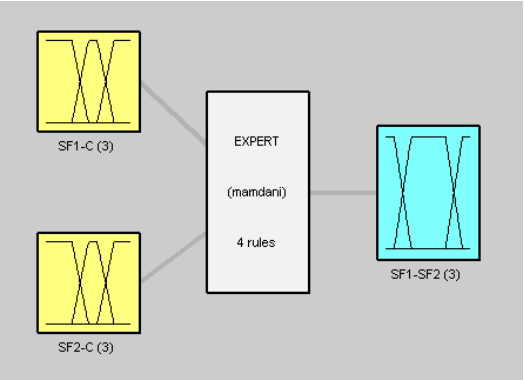


Figure IV-15. Architecture du traitement flou

Le traitement flou retenu est de type Mamdani [Dronkov et al., 1993] et comporte 3 étapes principales (figure IV-16) :

- Fuzzification des valeurs des interactions selon la modélisation adoptée pour les variables d'entrée (choix des variables linguistiques et de leurs ensembles flous);
- Utilisation des règles d'inférences pour caractériser la sortie par une des variables linguistiques ;
- Défuzzification de la sortie pour obtenir une valeur numérique.

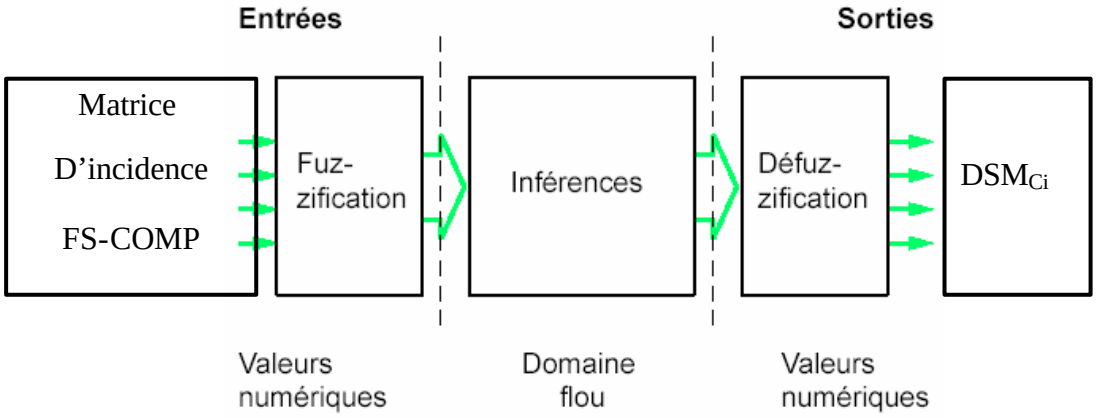


Figure IV-16. Les étapes du traitement flou

Concernant les autres caractéristiques du traitement flou adopté dans ce travail, la figure IV-17 nous donne un aperçu sur les plus importantes d'entre elles. En particulier, la méthode de défuzzification adoptée est celle du barycentre.

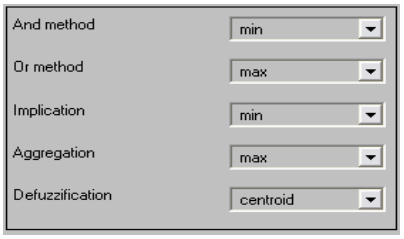


Figure IV-17. Caractéristiques du traitement flou

4.3.1. Caractérisation des variables d'entrée

Les MI sont été construites par les acteurs du projet. Nous leur demandons en plus de caractériser l'intensité des couplages en construisant des fonctions d'appartenance. Ces intensités appartiennent à]0, 10], nous avons alors opté pour le choix des variables linguistiques les plus utilisées à savoir: Faible, Moyen et Fort.

Arrivé à ce point, nous avons fait face à un dilemme concernant la construction de la variable linguistique Faible. Ce dilemme est lié au traitement à appliquer aux interactions nulles. En effet, deux possibilités s'offraient à nous :

- La première considère les interactions à valeur nulle comme des interactions non existantes et donc de ce fait la variable linguistique Faible ne couvre que les interactions à valeur non nulle.
- La deuxième considère que les interactions à valeur nulle existent. Alors le zéro est une valeur possible de la variable linguistique Faible.

Notre choix dans ce travail s'est portée sur la deuxième possibilité car:

- cette possibilité nous permet d'appliquer le traitement flou à toutes les valeurs de la matrice d'incidence,
- les valeurs nulles introduisent un biais dans le résultat du traitement flou, mais ce biais sera facilement corrigé par un filtrage permettant d'adapter les DSM, issues du traitement flou, à l'algorithme de *clustering*.

Concernant le choix du type des fonctions d'appartenance, nous avons opté pour les fonctions les plus utilisées à savoir les fonctions trapézoïdales.

Reprenons l'exemple de la MI FS-COMP. Sous l'éclairage de ces dernières hypothèses, toutes les FS interagissent à travers chaque composant C. On peut maintenant construire une $DSM_{Ci}FS$ pour chaque composant C_i.

La figure IV-18 montre les fonctions d'appartenance communes aux deux entrées. Ces fonctions ont été construites en coordination avec les concepteurs qui ont participé à la construction de la matrice d'incidence.

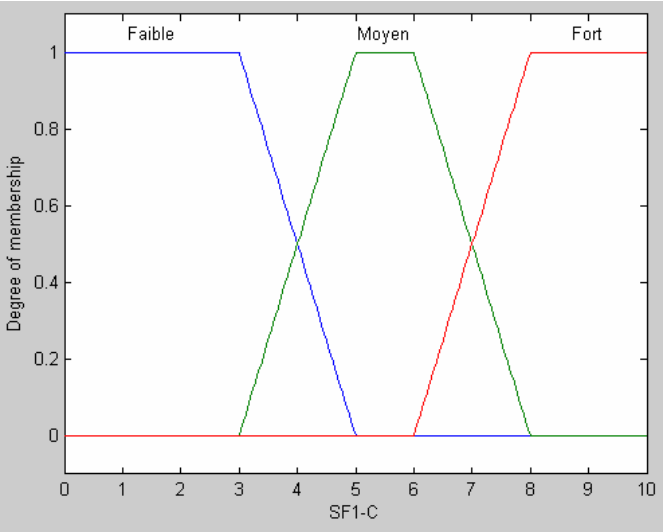


Figure IV-18. Les fonctions d'appartenance des variables d'entrée

Le choix des plages de définition de ces fonctions est motivé par la volonté d'être sélectif vis-à-vis des interactions fortes et moyennes d'où le décalage apparent vers la droite.

4.3.2. Caractérisation de la variable de sortie

Nous avons opté pour la même structure et les mêmes variables linguistiques pour caractériser la sortie, à savoir le couplage entre FS_j et FS_k.

Cependant, le choix des domaines des définitions pour les trois variables linguistiques dépend du remplissage de la matrice d'incidence. En effet, nous cherchons à avoir un écart-type important entre les valeurs obtenues. Cela est possible en défuzzifiant les sorties faibles et fortes avec des centres de gravité éloignés.

Cette règle nous a permis de choisir les fonctions d'appartenance présentées dans la figure IV-19. On remarque que contrairement aux fonctions caractérisant les entrées, la sortie se caractérise par :

- une fonction « Faible » réduite qui permet de limiter au maximum la valeur résultante de la défuzzification par centre de gravité. En effet, en réduisant la surface de la fonction Faible, on obtient comme résultat pour des entrées nulles la valeur de 1,06.
- une fonction « Fort » réduite qui permet contrairement à la fonction faible d'avoir une résultante par centre gravité élevée de l'ordre de 8,9. Cette valeur est comparativement éloignée de celle de 1,06.
- une fonction « Moyen » très large centrée autour de 5.

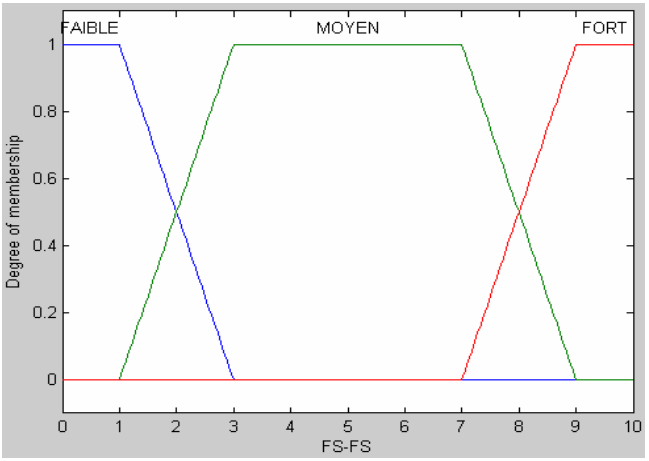


Figure IV-19. Les fonctions d'appartenance de la variable de sortie

4.3.3. Les règles d'inférences

Les règles d'inférences permettent de caractériser la variable de sortie d'une manière floue. Nous avons adopté dans ce travail pour les règles suivantes :

- SI (FS_j-C_i est *Faible*) OU (FS_k-C_i est *Faible*) ALORS (FS_j - FS_k est *Faible*)
- SI (FS_j-C_i est *Moyen*) ET (FS_k-C_i est *Moyen*) ALORS (FS_j - FS_k est *Moyen*)
- SI (FS_j-C_i est NON *Faible*) ET (FS_k-C_i est *Fort*) ALORS (FS_j - FS_k est *Fort*)
- SI (FS_k-C_i est NON *Faible*) ET (FS_j-C_i est *Fort*) ALORS (FS_j - FS_k est *Fort*)

La figure IV-20 montre une représentation en 3D de l'intensité de l'interaction entre FS_j - FS_k obtenue à travers les règles d'inférences et en fonction des caractérisations adoptées pour les variables d'entrée et de sortie.

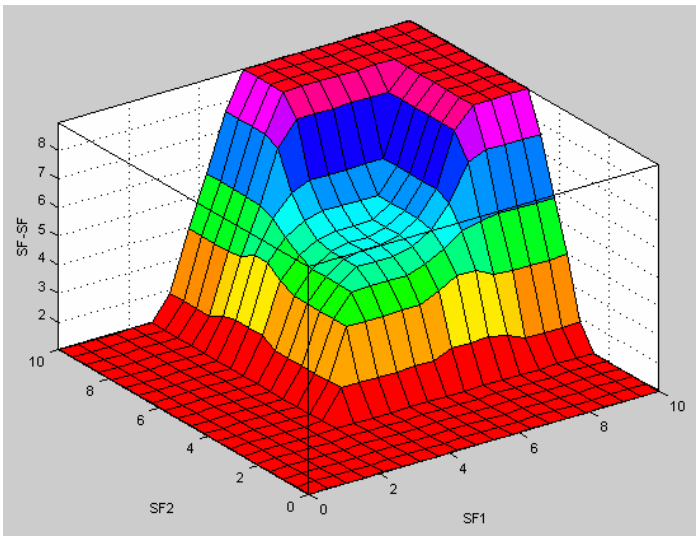


Figure IV-20. Représentation de la sortie en fonctions des deux entrées

La figure IV-20 nous permet d'analyser et justifier la combinaison des choix réalisés concernant les variables d'entrée, la variable de sortie et les règles d'inférence :

- Premièrement, les DSM obtenues vont faire l'objet d'un « clustering » pour identifier les modules et les éléments intégrateurs. L'algorithme de clustering utilisé est sensible à la densité des matrices et aux écarts entre les valeurs. Ceci justifie le choix de pentes avec une forte inclinaison.
- En cohérence avec l'hypothèse qu'on a prise concernant les interactions nulles, on remarque que pour deux valeurs d'entrée nulles et à travers la première règle d'inférence, nous obtenons pour la sortie une valeur non nulle (1.06) correspondant au barycentre du trapèze caractérisant la variable Faible.
- Si une fonction impacte faiblement un composant alors le couplage avec toute autre fonction qui impacte ce même composant ne peut être que faible. Au-delà du fait que cette affirmation est logique, on a voulu restreindre au maximum la propagation des couplages ayant une intensité faible (inférieures à 3). Ainsi la valeur minimale de la sortie (1.06) est atteinte pour 51% des couples des valeurs d'entrée. Ce choix aura pour conséquence finale de limiter la densité de la DSM résultante.
- Le palier (en cyan) représentant les sorties ayant des intensités moyennes n'est atteint que pour 4% des couples des variables d'entrée. Cette valeur est la plus faible par comparaison aux sorties ayant les intensités faibles et fortes.
- La sortie atteint son maximum pour 11% des couples des variables d'entrée.

On remarque alors que le traitement flou, tel que nous l'avons paramétré, favorise l'obtention d'intensités faibles en sortie plutôt que fortes. Ceci va nous permettre en utilisant le filtrage de réduire la densité des DSM générées.

La méthode de génération de la DSM résultante ainsi que le filtrage sont présentés dans les paragraphes suivants.

4.3.4. Génération de la DSM résultante

La méthode de traitement flou ainsi que les règles présentées précédemment nous permettent de générer une $DSM_{Ci}FS$ pour chaque composant C_i. En effet, si on considère une matrice d'incidence avec 13 fonctions et 15 composants, on obtient avec la méthode présentée 15 DSM FS. Cela représente le couplage entre les FS à travers chaque composant.

Pour obtenir la DSM résultante pour chacun des domaines traités, deux méthodes d'agrégation sont possibles :

- La méthode du maximum : si on considère $DSM(i,j)_{Ck}$ comme étant la valeur correspondant au couplage entre FS_i et FS_j dans la DSM générée à travers le

composant C_k, alors la DSM résultante notée DSM_X s'écrit : $DSM_X(i,j) = \max_k \ DSM(i,j)_{Ck}$.

- La méthode de la moyenne : la DSM résultante notée DSM_M s'écrit : $DSM_M(i,j) = \sum_{k=1}^{n} \frac{DSM(i,j)_{Ck}}{n}$ où n est le nombre de composants.

La comparaison entre ces deux méthodes se fera ultérieurement lors de l'application de la méthode à un exemple industriel.

4.3.5. Filtrage de la DSM résultante

Comme expliqué précédemment, l'un des inconvénients du traitement flou proposé est que les matrices obtenues sont denses avec l'impossibilité d'avoir des valeurs nulles même pour des interactions inexistantes. Etant donné que les matrices denses faussent les résultats attendus par l'algorithme de clustering, nous proposons dans ce paragraphe une méthode de filtrage des données dans les DSM résultantes.

Le filtrage proposé agit comme un filtre passe haut, en annulant les valeurs les plus basses. L'Eq.IV-1 montre la formalisation que nous avons adoptée.

$$Si\ DSM(i,j) \leq X\ \ alors\ DSM(i,j) = 0 \qquad \textbf{Eq.IV-1}$$

La valeur du seuil *X* est à fixer en fonction de la densité de la matrice recherchée.

Le filtrage ne sera pas explicité à chaque fois que l'on génère une DSM dans notre travail. Il faut cependant garder en mémoire qu'il est systématiquement réalisé pour diminuer la densité de la matrice et corriger le biais apporté par le traitement flou.

La détermination de la valeur du seuil est fixée par l'utilisateur, mais elle peut être automatisée en prenant en compte les trois contraintes suivantes :

- Si la densité de la DSM agrégée est supérieure à 70%, augmenter progressivement le seuil tout en respectant les deux conditions suivantes.
- Ne pas annuler toutes les valeurs d'une ligne ou d'une colonne.
- Appliquer un filtrage avec un seuil de préférence inférieur à 3 et qui peut être au maximum de 5 (valeur maximale pour caractériser une interaction faible).

La figure IV-21 montre le résultat du filtrage sur une DSM avec comme seuil X=2.9 et une densité inférieure à 70%.

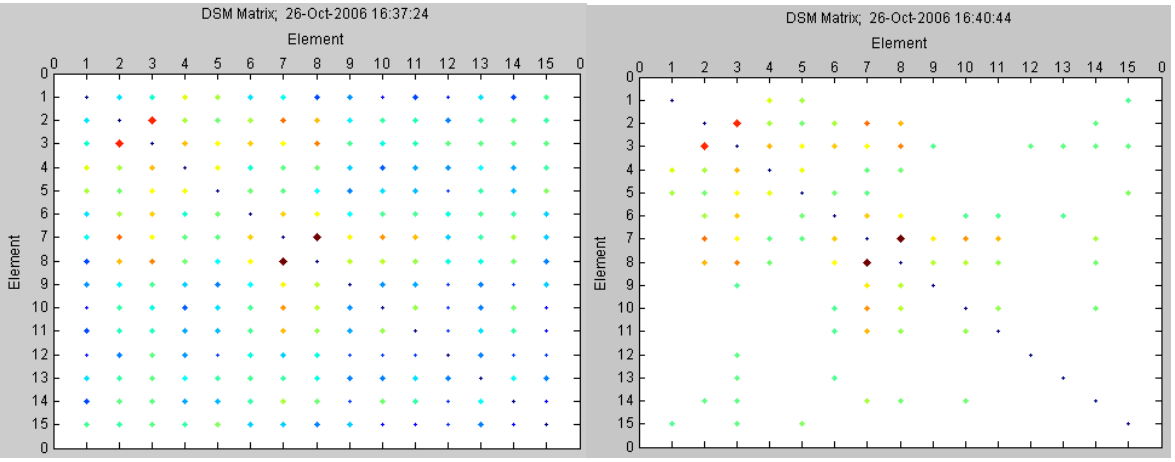


Figure IV-21. Illustration du filtrage sur les DSM

Nous venons de justifier la nécessité de réaliser un filtrage. Cependant, nous devons aussi vérifier que ce filtrage n'altère pas l'identification de l'architecture. Nous avons identifié deux cas où le filtrage pourrait influencer les résultats attendus :

- Le premier cas correspond à un élément qui interagit avec un grand nombre d'éléments mais avec des intensités faibles. Le filtrage va réduire le nombre d'interactions de cet élément et il va perdre de ce fait son caractère intégrateur. Cependant, nous avons prévu dans l'algorithme de clustering de permettre à l'utilisateur de désigner directement les éléments qu'il pense être intégrateur. Ainsi, il est possible de corriger l'effet du filtrage correspondant à ce cas;
- Le deuxième cas concerne un élément -par exemple un composant- qui dans la matrice d'incidence est couplé à un nombre limité de fonctions. Cet élément peut se retrouver dans la DSM agrégée avec de faibles et peu nombreuses interactions. Cet élément n'étant pas intégrateur, il appartient nécessairement à un module. Nous avons caractérisé le filtrage de telle sorte qu'il n'annule aucune ligne et aucune colonne. De ce fait, cet élément aura toujours une interaction non nulle dans la DSM. Cette interaction est suffisante pour le faire appartenir à un module.

Nous venons de justifier, que même si le filtrage peut annuler certaines valeurs des interactions, l'objectif de notre travail qui est l'identification des architectures et non l'évaluation des interactions entre éléments, ne sera pas perturbé.

4.4. Application à la MIFS-COMP

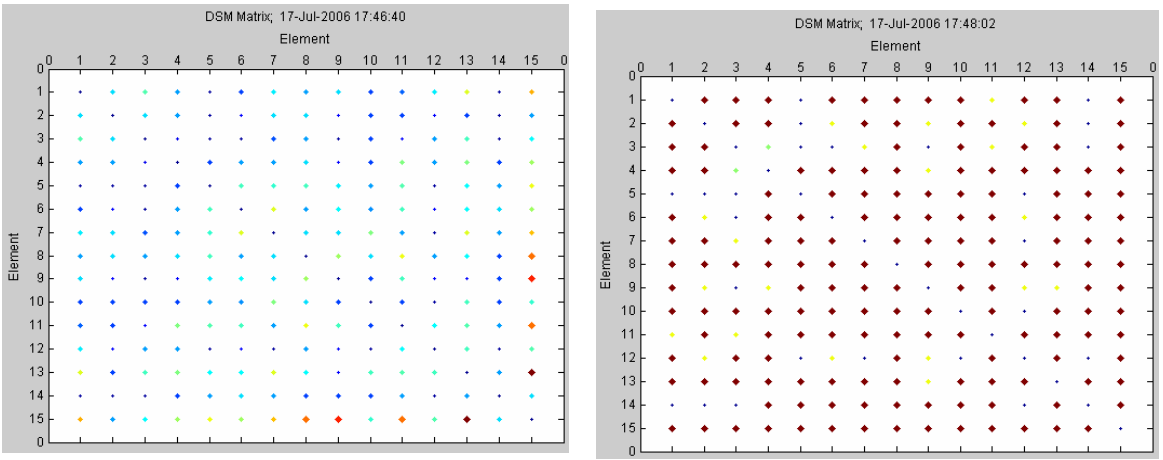
Nous présentons dans ce qui suit, l'application de la méthode sur l'exemple de la matrice d'incidence FS-COMP.

4.4.1. Construction de deux DSM à partir de la MI FS-COMP

La matrice d'incidence utilisée pour architecturer les domaines du produit est présentée dans la figure IV-7. En partant de cette matrice, nous pouvons générer deux DSM, une pour chaque domaine. Rappelons cependant que le processus flou permet d'obtenir ces DSM par agrégation de plusieurs DSM intermédiaires. Nous avons proposé deux méthodes d'agrégations : par la moyenne ou par le maximum. Dans ce qui suit, nous allons comparer les deux méthodes d'agrégation proposées sur la base de la DSM résultante du domaine des composants.

4.4.1.1. Illustration de la méthode d'agrégation

Les deux DSM obtenues sont représentées, d'une manière graphique, dans les figure IV-22(a) et IV-22(b). Nous notons DSM-moy, la DSM agrégée par la méthode de la moyenne et DSM-max, celle obtenue par la méthode du maximum. La figure IV-23 montre, dans leur forme numérique, les DSM calculées.



(a) DSM obtenue par la méthode la moyenne (b) DSM obtenue par la méthode du maximum

Figure IV-22. Comparaison des deux méthodes d'agrégation

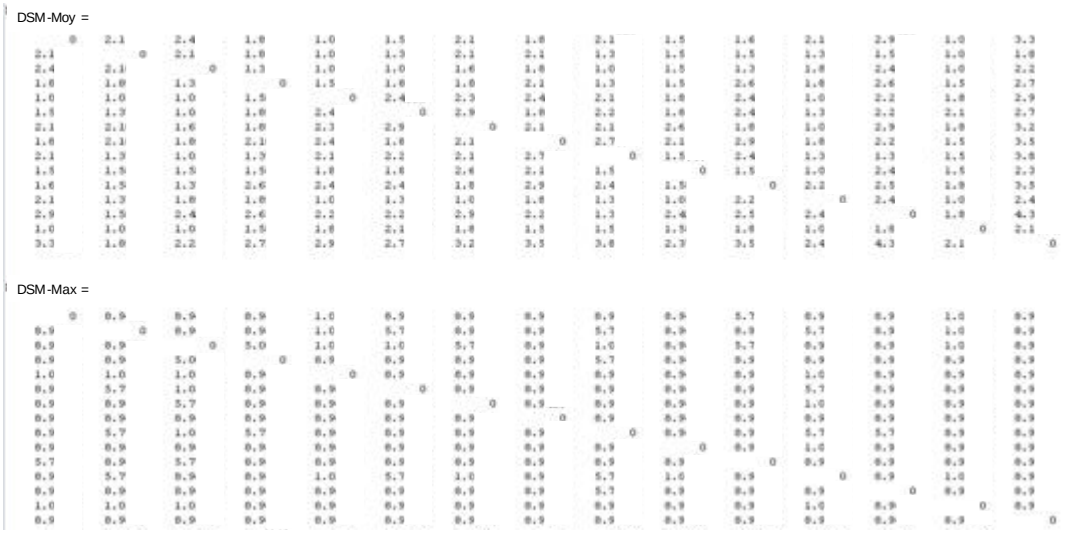


Figure IV-23. Forme numérique des deux DSM simulée

Nous remarquons que les matrices générées à travers le traitement flou sont évidemment sans valeurs nulles (nous rappelons que les éléments sur la diagonale n'ont aucun sens). Cependant, on peut remarquer visuellement sur la représentation graphique des matrices que :

- La méthode du maximum génère des valeurs relativement homogènes, nous dénombrons trois valeurs différentes (1.06, 5.77, 8.94) mais avec un grand nombre de 8,9, ce qui signifie que cette méthode est peu discriminante.
- La méthode de la moyenne génère des valeurs très différentes s'échelonnant de 1.06 à 4.36.

Le constat est que si la méthode du maximum est celle qui fournit le plus grand écart, elle est aussi celle qui uniformise le plus les valeurs des interactions, contrairement à la méthode des moyennes qui certes rend des valeurs dans une plage plus restreinte mais ces valeurs ont une plus grande variabilité.

En anticipant sur l'influence de ces données sur la qualité du processus d'identification des architectures par clustering que nous appliquons aux DSM obtenues, nous choisissons la méthode de la moyenne qui donne plus de variabilité aux valeurs des interactions.

La comparaison entre la méthode de la moyenne et la méthode du maximum peut se faire aussi par l'interprétation du principe de calcul de chaque méthode :

- L'intensité d'une interaction entre deux fonctions, calculée par la méthode du maximum, nous informe qu'il existe au moins un composant qui associe ces deux fonctions. Il suffit alors d'un seul composant pour que le couplage entre ces deux fonctions soit fort.
- L'intensité d'une interaction entre deux fonctions, calculée par la méthode de la moyenne, prend en compte le nombre de composants qui participent à la création du couplage entre les deux fonctions.

En prenant en compte l'influence de toutes les fonctions pour caractériser le couplage entre deux composants, la méthode de la moyenne se rapproche beaucoup plus de nos attentes vis-à-vis de la formalisation des règles de construction présentées au paragraphe 2.1.2 de ce chapitre. La méthode du maximum en ne prenant pas en compte la fréquence d'occurrence d'un couplage ne permet pas de distinguer l'influence réelle des éléments d'un domaine sur les interactions dans un autre domaine.

Dans la suite de ce travail, nous utilisons exclusivement la méthode de la moyenne pour générer les DSM résultantes. Les DSM ainsi obtenues doivent être filtrées pour limiter leur densité.

4.4.2. Identification des architectures fonctionnelle et organique

Après avoir construit les DSM résultantes par la méthode de la moyenne, nous allons dans ce paragraphe filtrer les DSM et proposer une architecture pour les domaines fonctionnel et organique du moteur. L'identification de l'architecture est réalisée par l'algorithme de clustering que nous avons développé dans le chapitre III.

L'algorithme de clustering nécessite le réglage des paramètres de fonctionnement. Le tableau IV-2 résume les réglages indépendants des caractéristiques de l'exemple. Ces réglages sont les mêmes pour toutes les applications.

Paramètres	Valeur
exp_taille	1
exp_int	2
qualité	4

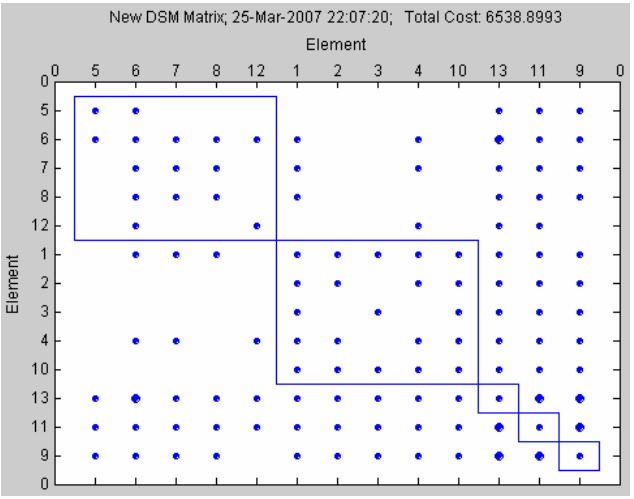
Tableau IV-2. Réglage des paramètres de l'algorithme

Deux autres paramètres de la méthode d'identification, à savoir IC (Indice de Couplage) et le seuil de filtrage, dépendent de l'exemple traité et seront précisés lors de la présentation de chaque application.

4.4.2.1. Génération de l'architecture fonctionnelle

L'algorithme de clustering utilise en entrée une DSM filtrée et propose une architecture de cette DSM. Le choix de l'architecture se fait sur la base de la minimisation d'une fonction coût. Comme nous l'avons précisé dans le chapitre III, l'algorithme peut ne pas converger à chaque fois vers la même architecture, on peut obtenir des architectures avec des coûts supérieurs au minimum déjà obtenu. Cependant, nous avons optimisé notre algorithme pour avoir une haute fréquence de répétition de l'architecture ayant le coût minimal. Dans la suite de ce travail et à chaque fois que l'algorithme de clustering est utilisé, l'architecture présentée est celle qui a réalisé le coût de couplage minimal et qui est en même temps la plus fréquemment obtenue.

En appliquant un filtrage de 2.6 à la DSM FS, nous obtenons une DSM (figure IV-24) avec une densité de 65% (102 interactions pour (169-13) interactions possibles). En partant de la DSM FS ainsi filtrée, l'algorithme de clustering, avec IC égal à 0.9, simule l'architecture représentée sur la figure IV-24.



1	2	3	4	5	6	7	8	9	10	11	12	13
RESP	ALIM CARB	DEPOL	COMB	CPC	PLV	EFF	CES	ACV	IES	THE	COU	VOL FONC

Figure IV-24. Architecture fonctionnelle du moteur

4.4.2.2. Interprétation de l'architecture fonctionnelle

Après avoir généré cette architecture, nous avons demandé à un architecte système de l'analyser et de valider sa cohérence. Cette architecture présente les caractéristiques suivantes :

- On obtient deux modules de taille comparable et 3 éléments intégrateurs.
- Le premier module est composé de 5 Fonctions Systèmes : Respiration, Alimentation carburant, Dépollution, Combustion et Sensorique (IES). Les 4 premières fonctions forment la phase de combustion et de précombustion. La FS sensorique regroupe la fonction de commande et de détection (capteurs). L'algorithme de clustering propose de rapprocher la sensorique de la phase de précombustion, ce rapprochement est validé par les architectes systèmes qui confirment le grand besoin en contrôle de cette phase du fonctionnement du moteur.
- Le deuxième module est lui aussi composé de 5 fonctions : Conversion pression-couple (CPC), "Perte par frottement, Lubrification, Ventilation" (PLV), Reprise d'effort (EFF), Conversion énergie secondaire (CES) et Couplage (COU). Toutes ces fonctions appartiennent à la phase de postcombustion. Ces fonctions forment une chaîne mécanique qui transforme l'énergie thermique obtenue par la combustion, en énergie mécanique au niveau de la FS CPC à la boîte de vitesse (FS COU).
- Les deux modules sont couplés. Ces couplages s'expliquent par la particularité du fonctionnement du moteur qui utilise les FS mécaniques pour mettre en œuvre les FS de précombustion, ces couplages reposent sur des contraintes liées au choix de l'architecture physique (distribution, entraînement accessoires).

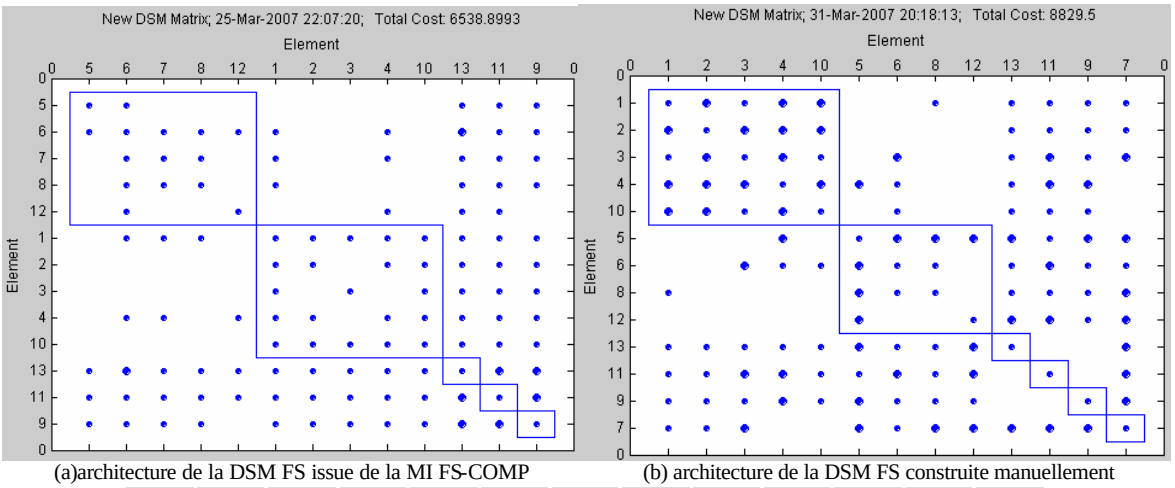
- Les FS intégratrices sont au nombre de 3 : la FS Volumes fonctionnels, la FS Acoustique-vibratoire et la FS Thermique. La première FS est « chargée » d'allouer les volumes physiques à chaque composant du moteur, elle est de ce fait intégratrice parmi les composants et donc parmi les FS. La deuxième FS est « chargée » de caractériser chaque composant du point de vue vibratoire, elle aussi est de ce fait intégratrice. Enfin, la FS Thermique caractérise et propage les contraintes thermiques entre les composants, cette fonction est donc intégratrice.

Pour résumer, l'architecture identifiée par l'algorithme de clustering repose sur deux modules de fonctions qui correspondent aux deux grandes phases du fonctionnement du moteur, à savoir la précombustion et la postcombustion. Ces deux phases même si elles sont connues par les architectes systèmes, n'ont pas, jusqu'à maintenant, été utilisées pour structurer le domaine des FS du moteur.

L'analyse de l'architecture proposée par l'algorithme de clustering et sa comparaison par rapport à la logique du produit et des attentes des acteurs ne suffisent pas pour conclure quant à la pertinence de la méthode proposée. C'est pourquoi nous avons demandé aux acteurs du projet de conception du moteur de construire directement la DSM FS. Cette DSM sera appelée DSM FS construite ou attendue.

4.4.2.3. Comparaison des architectures obtenues à partir de la DSM FS simulée et de la DSM FS construite

Pour pouvoir analyser objectivement la pertinence de la méthode de construction des architectures et de l'algorithme de clustering, nous avons voulu comparer la DSM FS obtenue à partir de MI FS-COMP à une DSM FS construite directement par les acteurs du projet. La figure IV-25 montre côte à côte les deux DSM FS : (a) DSM FS issue de la matrice d'incidence FS-COMP, (b) DSM FS construite manuellement (introduite au § III.).



(a)architecture de la DSM FS issue de la MI FS-COMP (b) architecture de la DSM FS construite manuellement

1	2	3	4	5	6	7	8	9	10	11	12	13
RESP	ALIM CARB	DEPOL	COMB	CPC	PLV	EFF	CES	ACV	IES	THE	COU	VOL FONC

Figure IV-25. Comparaison de deux architectures fonctionnelles du moteur

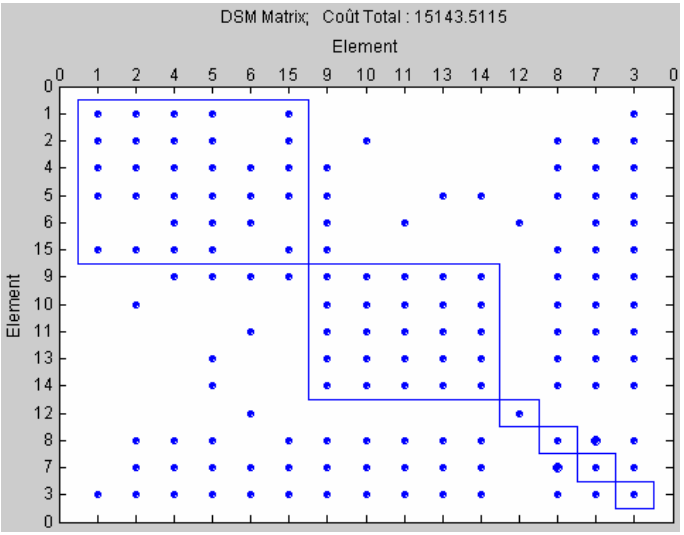
La comparaison des deux architectures suscite les remarques suivantes :

- Les deux architectures comportent deux modules de taille équivalente : l'un des modules est le même dans les deux architectures. Ce module est composé des fonctions Respiration, Alimentation Carburant, Dépollution, Combustion et Contrôle (IES). Ce module a été déjà identifié comme celui correspondant à la phase de précombustion et combustion du fonctionnement du moteur.
- Le deuxième module de l'architecture de la DSM FS attendue est inclus dans le deuxième module de la DSM FS simulée. Les fonctions en question sont : la Conversion Pression Couple (CPC), la Lubrification (PLV), la Conversion d'énergie secondaire (CES) et le Couplage (COU). Toutes ces fonctions font partie de la phase de postcombustion. Dans la DSM FS simulée, nous retrouvons en plus la FS EFF. Dans la DSM FS attendue, cette fonction est identifiée comme étant intégratrice par notre algorithme et ce, à cause du grand nombre de couplages qui lient cette fonction aux autres fonctions du moteur. En demandant l'arbitrage des acteurs du projet, ils concluent à l'importance de la FS EFF mais ils préfèrent la voir comme faisant partie du module de postcombustion.
- Les autres éléments intégrateurs sont en commun aux deux architectures. Il s'agit de l'Acoustique-vibratoire (ACV), la Thermique et la FS Volumes fonctionnels. Ces fonctions font consensus concernant leur caractère intégrateur.

Les deux architectures analysées ci-dessus sont très proches. Cela nous permet de conclure quant à la pertinence de la méthode de construction des architectures en partant d'une matrice d'incidence. Cependant, les processus de conception de PSA n'utilisent pas la conception modulaire et les acteurs du projet ne sont pas familiarisés avec les concepts que nous avons introduits dans le chapitre I partie 3. Or, l'identification des éléments intégrateurs est semi-automatisée dans notre algorithme de clustering et repose de ce fait en partie sur le jugement des acteurs du projet. Dans cet exemple ainsi que dans les exemples suivants, il faut garder en mémoire que l'identification des éléments intégrateurs est un consensus entre la perception des acteurs du projet et notre méthodologie de clustering.

4.4.2.4. Génération de l'architecture organique du moteur

L'algorithme de clustering appliqué à la DSM COMP génère l'architecture présentée dans la figure IV-26. Cette architecture a été obtenue avec un IC=0,65 et un seuil de filtrage à 2.2



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EGR	Injection	Culasse	Admis.	Echapp.	Distrib.	Attelage mobile	Carter	Lubrif.	Entraîn. Access.	Entraîn. Synchro.	Circuit de vide	Circuit calopor.	GSE.	CC

Figure IV-26. Architecture organique du moteur

Après clustering, l'architecture obtenue est analysée comme suit :

- L'algorithme de clustering a identifié 3 modules et 3 éléments intégrateurs ;
- Le premier module est composé des 6 constituants : EGR, Injection, Admission d'air, Echappement, Distribution et le système de Contrôle-Capteurs. Ce module correspond physiquement à la partie haute du moteur, sans la culasse qui est un composant intégrateur. Cette partie du moteur est le siège des phénomènes physico-chimiques et thermodynamiques, qui correspondent à la phase amont du fonctionnement du moteur où l'air et le carburant sont captés et acheminés à la chambre de combustion à travers les moyens d'injection. Cependant, nous retrouvons dans ce module, la Distribution qui achemine l'énergie mécanique vers la partie haute du moteur pour permettre son fonctionnement. Ce constituant est identifié par les acteurs du projet comme étant plus proche de la partie basse du moteur.
- Le deuxième module est composé de 5 constituants : la Lubrification, le circuit caloporteur, les deux Entraînements et le système de Gestion et stockage d'énergie (GSE). Tous ces constituants se trouvent en aval de la phase de combustion, pour transmettre et transformer l'énergie mécanique en énergie utile au fonctionnement propre du moteur (les deux entraînements et la GSE) et pour faciliter la transmission de l'énergie mécanique ou absorber les dissipations d'énergie par frottement (Lubrification).
- Dans l'architecture obtenue, le circuit du vide est un composant qui est relativement peu couplé aux autres composants du moteur. Cela s'explique par le nombre limité de FS qui interagissent avec ce composant. Le traitement flou, la méthode d'agrégation par la

moyenne et le filtrage font alors que les interactions entre le circuit du vide et les autres composants sont peu nombreuses et ont des intensités faibles. Selon les acteurs du projet, le circuit du vide peut être perçu comme un composant-module à cause de la complexité réduite de ce composant qui fait qu'il participe à une unique fonction du moteur : créer du vide.

- Les trois composants intégrateurs du moteur sont la culasse, l'attelage mobile et le carter :
  - La culasse est le constituant support de la partie haute du moteur. C'est sur ce composant que se fixent les éléments participant à la phase de précombustion et de combustion. Il est de ce fait clairement intégrateur.
  - Le carter, par symétrie avec la culasse, forme le composant support de la partie basse du moteur. La culasse et le carter sont les principaux composants réalisant la FS EFF qui est la fonction de protection et d'enveloppe du moteur.
  - L'attelage mobile, quant à lui, est le constituant d'interface entre les deux phases de fonctionnement du moteur. C'est à son niveau que l'énergie thermodynamique de la combustion se transforme en énergie mécanique. De ce fait, il joue le rôle de constituant intégrateur interne au moteur.

4.4.2.5. Architecture globale du Produit

La méthode de conception des architectures que nous proposons dans ce travail permet de capturer à la fois l'architecture d'un domaine et de ses sous-domaines. Dans les deux paragraphes précédents, nous avons montré comment passer d'une matrice d'incidence FS-COMP à l'identification de l'architecture des fonctions systèmes et des constituants du produit. A partir de ces trois matrices (2DSM et une MI) nous pouvons aussi construire la DSM globale du produit présentée sur la figure IV-27.

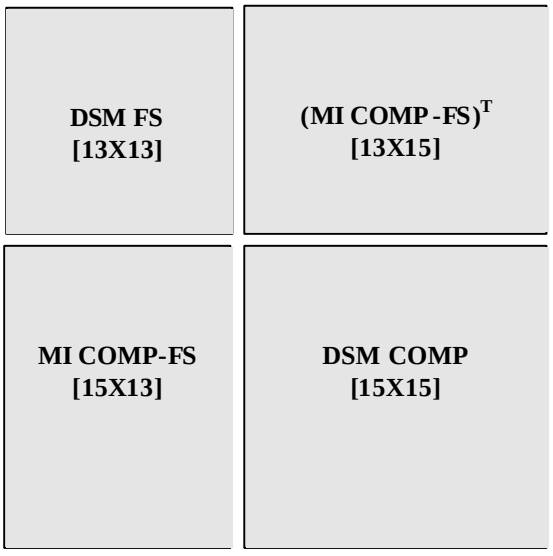
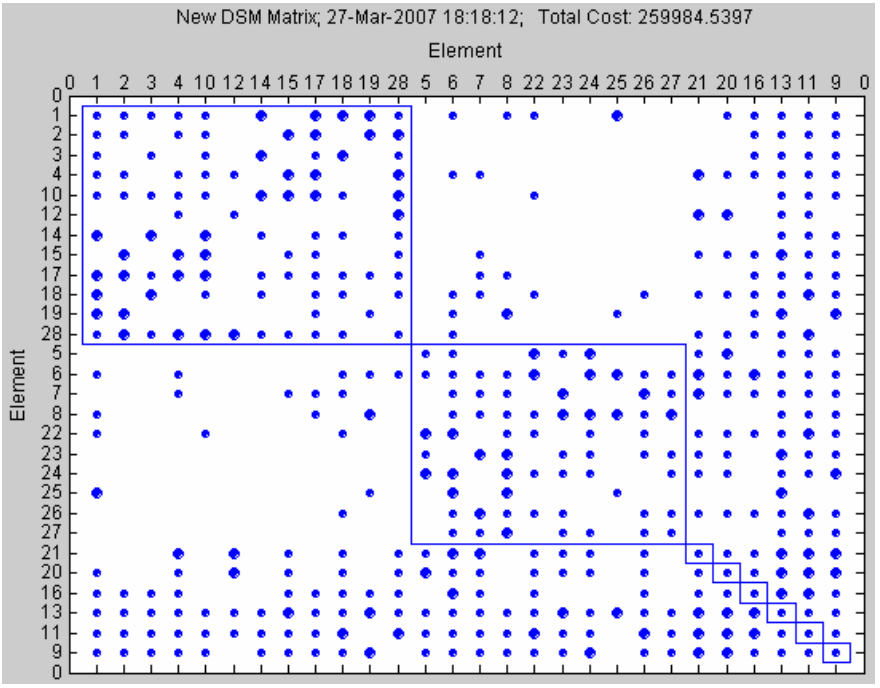


Figure IV-27. Modélisation de la DSM du produit

La matrice obtenue est de dimension 28, il s'agit bien de la DSM P (Produit) étant donné que les éléments en diagonale sont nuls et les valeurs de toutes les interactions sont comprises entre 0 et 10. Cette DSM P peut être utilisée directement en entrée de l'algorithme de clustering pour identifier l'architecture globale du produit. Cette démarche va nous permettre de fournir aux architectes systèmes des modules cohérents dont les fonctions sont couplées prioritairement aux composants qui appartiennent au même module. Si cette cohérence est validée, il est possible d'identifier, avec des interfaces physiques et fonctionnelles bien définies, des modules du moteur qui peuvent devenir des "modules sur étagère".

La figure IV-28 montre l'architecture optimale identifiée par l'algorithme de clustering avec un IC=0.9



1	2	3	4	5	6	7	8	9	10	11	12	13
RESP	ALIM CARB	DEPOL	COMB	CPC	PLV	EFF	CES	ACV	IES	THE	COU	VOL FONC

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
EGR	Injection	Culasse	Admis.	Echapp.	Distrib.	Attelage mobile	Carter	Lubrif.	Entraîn. Access.	Entraîn. Synchro.	Circuit de vide	Circuit calopor.	GSE.	CC

Figure IV-28. Architecture globale du moteur

L'architecture obtenue suscite les remarques suivantes :

- L'architecture globale du produit se compose de deux modules de taille comparable (12 et 10) et de 6 éléments intégrateurs ;
- Le premier module du produit correspond, à un élément près, à la réunion du modules de fonctions {1, 2, 3, 4, 10} et celui de constituants {14, 15, 17, 18, 19, 28}, ces modules correspondant à la phase de précombustion et de combustion. Nous retrouvons en plus de ces éléments la fonction couplage {12}. Le fait de retrouver la fonction de

couplage dans ce module peut s'expliquer par la nécessité de lier très tôt dans le processus de conception les caractéristiques du couplage avec les caractéristiques de la phase de précombustion. La phase de post-combustion a surtout la fonction de transmettre l'énergie créée et de fournir l'énergie propre au fonctionnement du moteur.

- Le deuxième module correspond en majorité à la réunion de deux modules précédemment identifiés : le module de fonctions {5, 6, 7, 8}, le module de constituants {22, 23, 24, 25, 26, 27} participant à la réalisation de la phase de postcombustion du moteur. Nous pouvons remarquer l'introduction du constituant "circuit de vide" dans ce module. La composition de ce module est tout à fait cohérente et correspond aux attentes des acteurs du projet.

- Les éléments intégrateurs sont au nombre de 6. Ce sont les éléments intégrateurs identifiés dans l'architecture des fonctions et des constituants. A côté de ces éléments intégrateurs, nous avons identifié deux éléments qui sont éligibles pour être intégrateurs, sauf que les interactions qu'ils mettent en œuvre s'adressent majoritairement au deuxième module. Ces deux éléments sont les fonctions système PLV et Reprise d'effort.

A travers l'analyse qu'on vient de réaliser de l'architecture globale du moteur, nous pouvons remarquer la grande complexité de ce système avec des éléments (fonctions et composants) très fortement couplés entre eux. La démarche d'identification des architectures appliquée au moteur en partant de la MI FS-COMP montre qu'on identifie uniquement deux modules. Ces deux modules sont relativement grands (avec de nombreuses interfaces) pour que l'on puisse envisager d'en faire des modules physiques réels. Cependant, nous pensons que notre démarche peut aider à améliorer des processus de conception et à surmonter la complexité du moteur. Notre méthode a montré que dans le processus de décomposition et de spécification, il est intéressant de passer par une étape dans laquelle on définit les deux modules et les six éléments intégrateurs identifiés dans l'architecture. Cette étape supplémentaire va permettre de spécifier et de figer les interfaces externes puis ensuite les interfaces internes des modules, et en parallèle, de caractériser et d'intégrer l'ensemble en concevant les éléments intégrateurs.

4.5. Conclusions

La méthode permettant de construire deux DSM à partir d'une MI et d'identifier leurs architectures a fourni des résultats pertinents selon l'interprétation et l'avis des architectes systèmes. En effet, on retrouve le même principe d'architecture autour des phases de précombustion et de postcombustion. Cependant, nous pensons que ces architectures peuvent être améliorées en considérant les autres situations de conception.

5. Troisième situation : L'architecture des domaines face à la redondance

5.1. Principe

Dans la deuxième situation, nous avons montré qu'il est possible de construire à partir d'une seule MI deux DSM. Il est possible alors d'avoir deux MI ayant un domaine en commun. Dans ce cas, nous obtenons 4 DSM avec 2 DSM pour ce domaine en commun.

Dans l'application illustrant la deuxième situation, nous avons utilisé une MI FS-COMP pour générer une DSM FS et une DSM COMP. Ensuite nous avons comparé la DSM FS à une DSM construite manuellement. Nous étions en présence alors d'une situation de redondance avec deux DSM pour un même domaine.

A travers les deux exemples précédents, nous avons réfléchi à une méthode qui profite de la situation de redondance pour enrichir et affiner les architectures qu'on simule. Nous modélisons la situation de redondance (figure IV-29) par l'existence de deux DSM d'un même domaine mais provenant de sources différentes. L'objectif est alors de proposer une nouvelle DSM à partir de ces deux DSM.

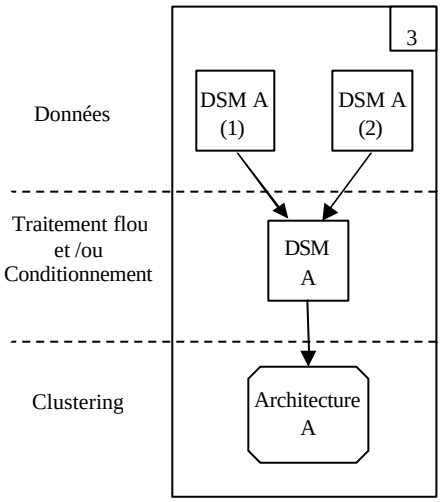


Figure IV-29. Schéma de la troisième situation

Les DSM sont des matrices, et donc des objets mathématiques, qui peuvent être manipulées par l'algèbre. Nous proposons d'utiliser une combinaison linéaire pour générer la matrice résultante. On écrit alors :

$$DSM = \sum_{i=1}^{n} a_i DSM_i \qquad \sum_i a_i = 1$$

Eq. IV-2

Il s'agit en réalité d'une pondération entre les DSM disponibles pour créer la DSM résultante. Cependant, les matrices que l'on combine ont des origines différentes. De ce fait, il faut respecter certaines règles lors de l'utilisation de la méthode de pondération :

- Certaines DSM redondantes peuvent avoir été construites directement par les acteurs du projet. Dans ce cas, et si on suppose que le guide de construction des DSM a été respecté (cf §2.1.), ces DSM prennent leurs valeurs entre 0 et 10. Ces DSM sont utilisables directement dans la combinaison linéaire.
- D'autres DSM peuvent être le résultat d'un traitement flou (situation 2 et 4). L'utilisation de la méthode de la moyenne pour construire les DSM finales fait que les interactions les plus fortes en sortie ne sont pas égales à 10. Pour que ces DSM puissent être comparables à celles construites manuellement, nous proposons de les normaliser avec comme valeur maximale 10.

En rendant les DSM homogènes et comparables, il est possible d'utiliser la combinaison linéaire pour proposer la DSM résultante. L'algorithme de clustering qui permet d'identifier l'architecture sous-adjacente du domaine est alors appliqué sur la matrice résultante.

Dans ce qui suit, nous allons étudier sur la base d'un exemple la mise en œuvre de la méthode de résorption de la redondance et les avantages qu'elle présente.

5.2. Mise en œuvre

Pour illustrer la méthode de résorption de la redondance, nous allons reprendre l'exemple du projet de conception du moteur diesel où à la fois, la matrice d'incidence FS-COMP et la matrice d'incidence EX-FS sont utilisées comme données.

La matrice d'incidence EX-FS est une matrice d'incidence numérique construite par les acteurs du projet selon les mêmes principes qui ont régi la construction de la matrice FS-COMP.

La figure IV-30 montre la MI numérique EX-FS avec, en colonne, les sept exigences vis-à-vis du moteur et en ligne les treize fonctions systèmes présentées précédemment.

SF/EX	Performance	Démarrage	Emission	Agrément	Vibration	Consommation	Fiabilité	Sécurité	Maintenabilité
Resp	9	8	9	8	7	8	9	7	8
Alim carb	8	9	9	8	5	8	9	8	8
Dépol	7	8	9	5	4	8	8	6	8
Comb	0	5	8	5	8	8	3	9	6
CPC	8	5	5	0	5	5	9	8	4
PLV	6	8	0	0	8	7	8	8	7
EFF	0	8	0	0	8	5	7	5	0
CES	5	5	0	0	0	8	6	9	6
ACV	5	0	0	0	8	8	5	3	5
IES	8	8	0	0	5	0	8	8	8
THE	8	0	8	8	8	8	6	5	5
COU	9	0	5	8	6	5	6	9	6
Vol Fonc	5	5	5	5	5	5	3	3	3

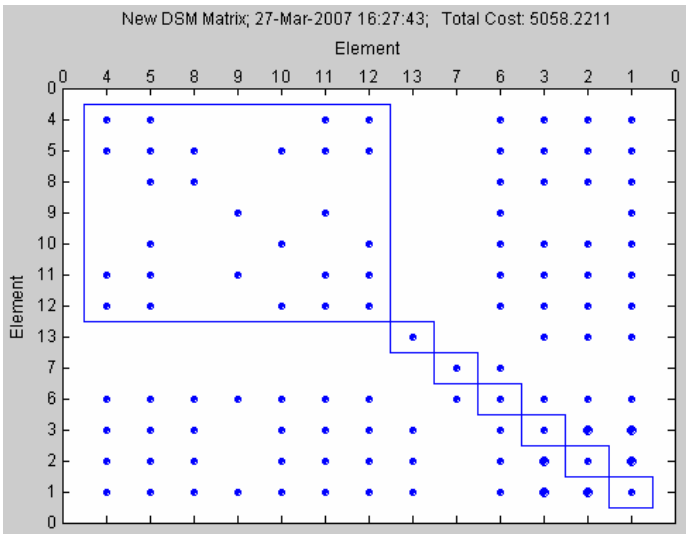
Figure IV-30. MI EX-FS

A partir de la matrice d'incidence EX-FS, nous pouvons générer par le processus flou présenté dans la deuxième situation les deux DSM correspondantes à chacun des domaines. Ensuite, à l'aide de l'algorithme de *clustering*, nous identifions l'architecture liée à chaque DSM.

Etant donné que notre intérêt se porte sur la résorption de la redondance, nous allons axer notre travail sur les deux DSM FS obtenues.

5.2.1. Génération de la DSM FS à partir de la MI EX-FS

La DSM résultante a été générée selon la méthode de la moyenne avec un seuil de filtrage $X_M = 4.3$ et un IC=0.8.



1	2	3	4	5	6	7	8	9	10	11	12	13
RESP	ALIM CARB	DEPOL	COMB	CPC	PLV	EFF	CES	ACV	IES	THE	COU	VOL FONC

Figure IV-31. Architecture de la DSM FS issue de la MI EX-FS

L'architecture (Figure IV-31) identifiée nous amène à faire les remarques suivantes :

- L'architecture identifiée par l'algorithme de clustering montre une architecture déséquilibrée, avec un grand module composé de 7 FS, deux FS seules et 4 FS intégratrices ;
- La DSM obtenue nous renseigne sur la criticité des FS vis-à-vis des exigences qui s'appliquent au moteur : ainsi les FS 1, 2, 3 et 6 sont celles qui sont les plus impactées par les exigences et à l'opposé, les FS 7 et 13 le sont moins.
- La DSM FS générée à partir de la MI EX-FS est totalement différente de celle générée de la MI FS-COMP, étudiée précédemment. Il est évident alors que l'architecture issue de la matrice d'incidence FS-COMP est celle qui représente le mieux l'architecture attendue du domaine des FS. Cependant, bien que les exigences ne soient pas

structurantes pour les FS, nous pensons qu'elles peuvent donner un nouvel éclairage sur l'architecture des FS.

5.2.2. Agrégation des deux DSM FS

La méthode d'agrégation des deux DSM FS étant une simple combinaison linéaire, nous obtenons la formulation suivante :

$$FS\ DSM_{ag} = a \times FS\ DSM_{/COMP} + b \times FS\ DSM_{/EX}$$
$$a + b = 1 \quad a, b \in]0,1[$$

Eq.IV-3

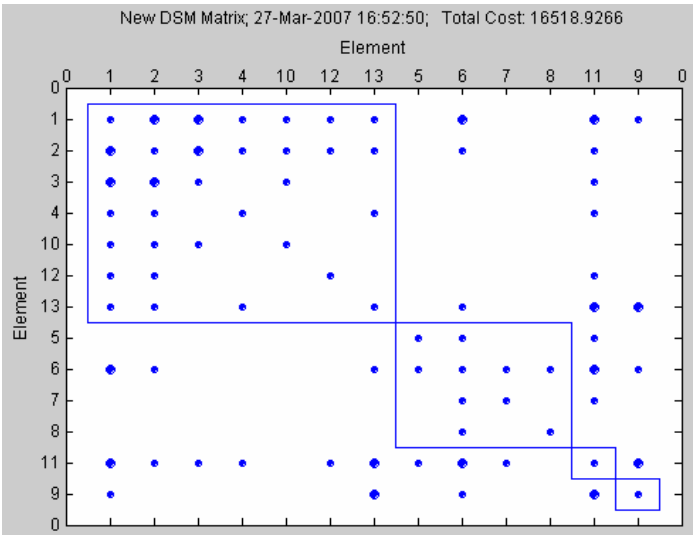
Où :

$FS\ DSM_{aq}$:	*La DSM résultante de l'agrégation*
$FS\ DSM_{/COMP}$:	*La DSM FS obtenue à partir de la matrice d'incidence FS-COMP*
$FS\ DSM_{/EX}$:	*La DSM FS obtenue à partir de la matrice d'incidence FS-EXP*

Le choix des valeurs des paramètres a et b peut être guidé par la réflexion suivante. Les interactions dans les MI utilisées dans ce travail sont toutes définies de telle manière qu'on peut générer d'une manière symétrique les DSM des deux domaines liés par cette MI. Dans ce cas, tous les domaines ont la même importance et aucune hiérarchie n'existe entre les domaines. Ainsi, logiquement on a : a = b = 0.5

Cependant, il est possible de considérer a différent de b, si on a plus confiance dans une méthode de construction ou dans les valeurs d'une DSM.

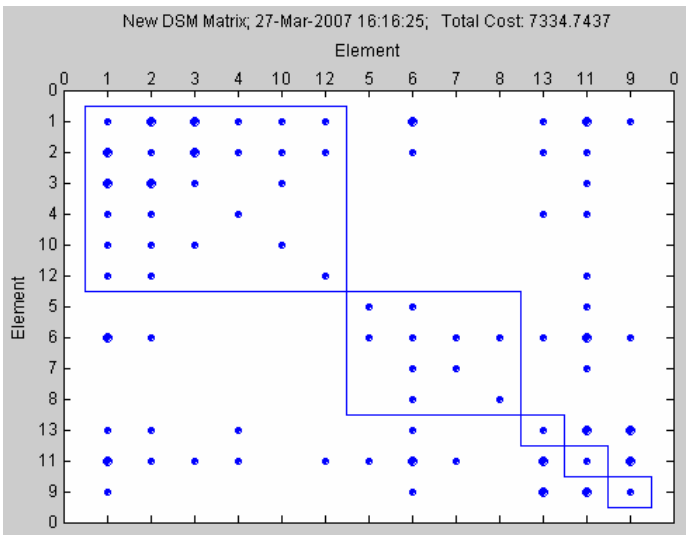
La figure iv-32 montre le clustering obtenu sur la matrice agrégée avec a=b=0.5 et où le seuil de filtrage est fixé à $X_m = 2.6$ et IC est égale à 0.8



1	2	3	4	5	6	7	8	9	10	11	12	13
RESP	ALIM CARB	DEPOL	COMB	CPC	PLV	EFF	CES	ACV	IES	THE	COU	VOL FONC

Figure IV-32. Architecture de la DSM FS agrégée

Nous remarquons dans la figure IV-32 que l'architecture identifiée comprend deux éléments intégrateurs (THE et ACV). Etant donné que seule la matrice d'incidence FS-COMP a permis d'obtenir une architecture admissible, nous décidons de désigner directement la FS VOL FONC comme étant intégratrice, on obtient alors l'architecture représentée dans la figure IV-33.



1	2	3	4	5	6	7	8	9	10	11	12	13
RESP	ALIM CARB	DEPOL	COMB	CPC	PLV	EFF	CES	ACV	IES	THE	COU	VOL FONC

Figure IV-33. Architecture de la DSM FS agrégée avec FS VOL intégrateur

Nous analysons alors l'architecture obtenue par comparaison à celle issue de la matrice d'incidence FS-COMP (Figure IV-24) :

- On obtient sensiblement deux architectures très proches, la différence principale réside dans le fait que la FS COU est passée du module postcombustion au module de combustion et précombustion, le premier module ainsi constitué devient le groupement des FS qui sont aussi les plus impactées par les exigences.
- L'enseignement que nous pouvons tirer à partir de l'architecture obtenue est que les FS de la précombustion outre le fait de constituer un module homogène sont les plus importants vis-à-vis de la satisfaction des exigences, et pour accomplir cette tâche, il faut prendre en considération la FS couplage dès les phases amont de la conception.

Pour conclure, on peut considérer que les deux architectures représentées en figure IV-24 et IV-33 correspondent à deux architectures des FS captées à deux instants différents de la vie du projet. En figure IV-33, elle correspond à une phase transitoire quand on réalise la descente des exigences vers les FS. Quant à l'architecture représentée en figure iv-24, elle correspond à l'architecture des FS lorsqu'on ne prend en compte que l'architecture organique du moteur.

6. Quatrième situation : Génération d'une DSM à partir d'une MI et d'une DSM

6.1. Introduction

La méthodologie présentée dans cette thèse repose sur l'hypothèse que l'on peut générer de l'information sur l'architecture des domaines du projet même si les DSM des domaines ne sont pas accessibles. Ainsi, dans un premier temps, nous avons proposé de générer les DSM en partant des MI. Dans un second temps, nous avons montré que l'agrégation de DSM permet d'aboutir à des architectures qui se rapprochent beaucoup plus fidèlement de la réalité du projet.

Dans ce paragraphe, nous allons envisager un nouveau cas de figure permettant la modélisation par DSM d'un domaine et l'analyse de son architecture. Ce cas de figure répond à certaines attentes des ingénieurs vis-à-vis de l'utilisation de la méthode de construction des architectures. En effet, nous avons présenté une méthode permettant de générer, à partir d'une matrice d'incidence, deux DSM et ainsi de caractériser en même temps l'architecture de deux domaines. La question qui se pose alors est : peut-on exploiter la situation où on a comme donnée la DSM d'un de ces domaines?

Nous proposons dans ce qui suit une méthode pour générer une DSM B connaissant MI A-B et DSM A (figure IV-34).

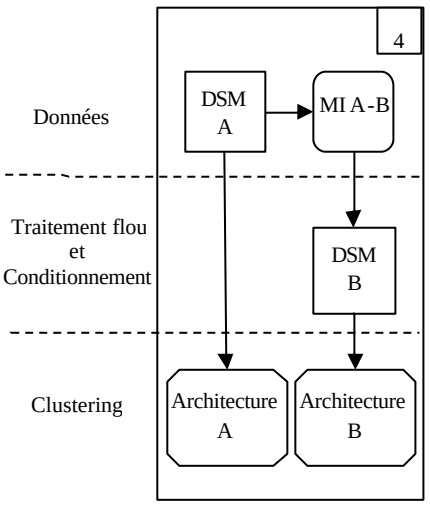


Figure IV-34. Schéma de la quatrième situation

Pour illustrer cette situation, nous allons reprendre l'exemple de la matrice d'incidence FS-COMP et nous allons utiliser comme donnée la DSM FS construite manuellement.

6.2. Principe

Notre objectif dans cette partie est d'utiliser une DSM A comme donnée d'entrée pour affiner l'architecture d'une DSM B, calculée à partir d'une matrice d'incidence A-B.

Nous avons opté pour l'utilisation d'un traitement flou qui permet de générer la DSM du domaine B. Ce traitement flou est la traduction de règles de construction qui permettent de lier les interactions inter-domaines (DSM A) et intra-domaines (MI A-B) pour identifier et évaluer les interactions dans la matrice B.

A l'image du traitement flou que nous avons proposé pour construire les DSM à partir de d'une matrice d'incidence, le traitement flou de cette nouvelle méthode repose sur des règles de construction.

6.2.1. Les règles de construction

Nous allons conserver la même notation que celle utilisée dans le début de ce paragraphe avec deux domaines : A (composé d'élément A_i) et B (composé d'élément B_i) et deux matrices : une matrice d'incidence A-B (représentant les interactions A_i-B_j) et une DSM A (représentant les interactions A_i-A_j).

Les règles de construction de la DSM B sont les suivantes :

Règle 1. Lorsqu'un couple d'éléments (A_i, A_j) est couplé à un couple (B_u, B_v) alors soit A_i est couplé à B_u et A_j est couplé B_v, soit A_i est couplé à B_v et A_j est couplé B_u.

Règle 2. Soit B_u et B_v deux éléments de B, s'il existe deux éléments A_i et A_j qui interagissent dans la DSM A et si en même temps le couple d'éléments (A_i, A_j) est couplé au couple (B_u, B_v) alors B_u et B_v sont couplés.

Règle 3. L'intensité du couplage entre B_u et B_v dépend de l'intensité du couplage entre A_i et A_j et des intensités de couplage entre les couples (A_i, A_j) et (B_u, B_v).

Règle 4. Nous considérons qu'un élément est couplé à lui-même avec une intensité égale à 10.

Nous allons expliquer maintenant les règles que nous avons retenues :

- La première règle précise que lorsque dans une matrice d'incidence on considère un couple d'éléments appartenant à un domaine A (A_i, A_j) et un autre appartenant à B (B_u, B_v) alors il y a deux combinaisons possibles des couplages : soit A_i avec B_v et A_j avec B_u, soit A_i avec B_u et A_j avec B_v. Dans la figure IV-35, nous avons choisi de représenter A_i couplé avec B_u et A_j couplé avec B_v.
- La deuxième règle reflète l'idée que nous voulons propager les interactions d'un domaine A vers un domaine B. Ainsi le couplage entre deux éléments de B provient de l'existence de deux éléments de A qui sont couplés entre eux et qui interagissent respectivement avec chacun des éléments de B. Dans la figure IV-35, l'existence du couplage entre A_i et A_j implique l'existence d'un couplage entre B_u et B_v.
- La règle 3 reprend la règle 2 et est affirme que les intensités sont liées entre elles.

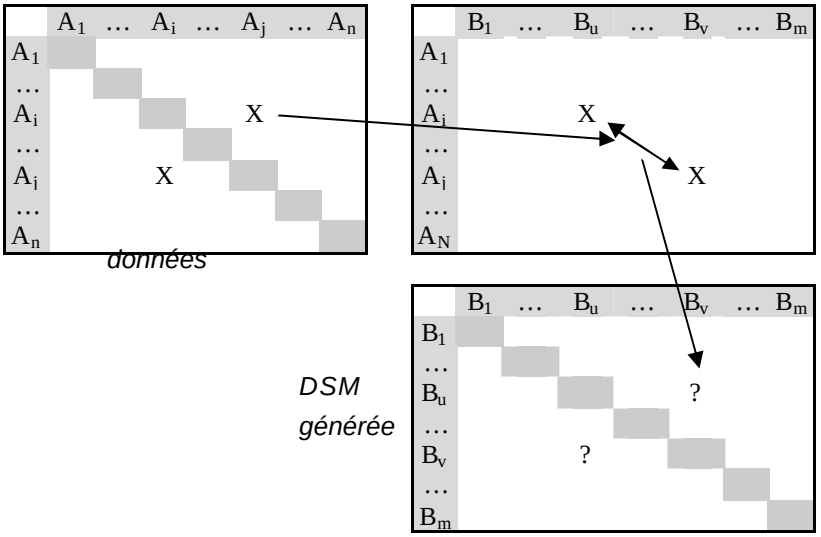


Figure IV-35. Propagation d'une interaction de A vers B

- La règle 4 introduit une extension du concept de DSM en considérant que les éléments en diagonal ont une valeur de 10. Cette règle nous permet de prendre en compte la possibilité de créer un couplage entre deux éléments de B qui sont couplés au même élément de A. Pour conserver la même formulation que dans le cadre général, on considère alors que A_i est couplé à lui-même avec une intensité de 10. La figure IV-36 montre l'application de cette règle.

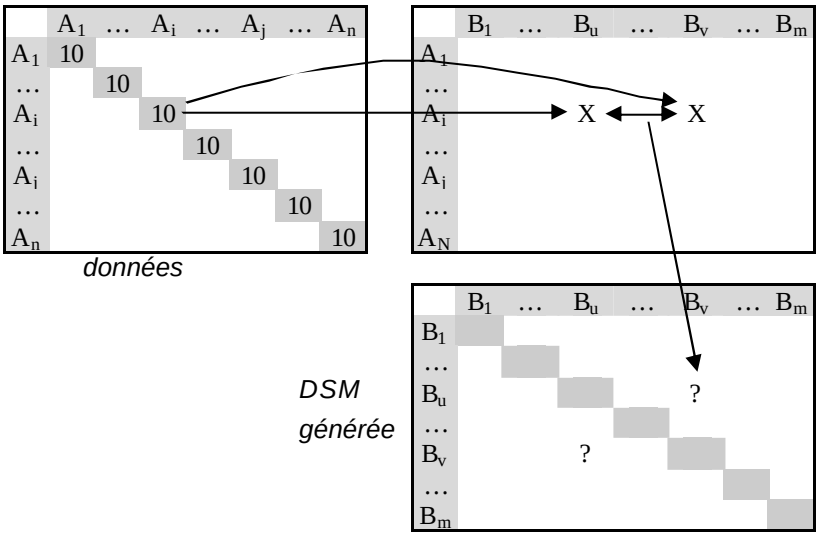


Figure IV-36. Propagation des intensités en diagonale de A vers B

6.2.2. Préparation des DSM pour le traitement flou

Les règles d'inférence présentées dans ce processus flou s'appliquent à des valeurs numériques allant de 0 à 10. Cependant, si la MI est toujours construite manuellement, la DSM utilisée en entrée peut être soit construite manuellement, soit obtenue par un algorithme (par exemple, celui qui permet de générer des DSM à partir d'une MI). C'est le deuxième

type de DSM qui peut poser problème. En effet, avec la méthode d'agrégation par la moyenne, les intensités ont un faible écart-type et l'interaction la plus forte peut être de 5. C'est pourquoi, nous proposons de normaliser les DSM en entrée pour avoir des valeurs allant de 0 à 10.

La normalisation se présente comme suit : soit DSM A, une DSM avec des valeurs quelconques positives alors :

$$DSMA_N = \frac{DSMA}{\max_{i,j}(DSMA(i,j))} \times 10 \qquad \textbf{Eq.IV-4}$$

La normalisation a alors pour effet d'étaler les valeurs d'entrée entre 0 et 10 tout en conservant le rapport entre les intensités.

6.2.3. Le processus flou

Comme nous l'avons précisé pour le premier processus flou, ce processus va permettre de formaliser la construction de la valeur des interactions en sortie en fonction des valeurs des interactions en entrée (figure IV-37).

Toute caractéristique de ce processus flou qui n'est pas précisé ici est inchangée par rapport au processus flou présenté dans la partie 4.3.

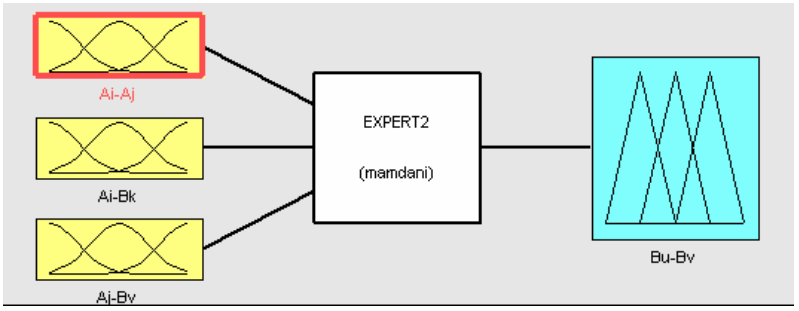


Figure IV-37. Architecture du traitement flou

Les variables d'entrée ainsi que la variable de sortie utilisent la même caractérisation basée sur 3 variables linguistiques : Faible, Moyen et Fort. Les fonctions d'appartenance sont les mêmes que pour l'autre traitement flou.

6.2.3.1. Les règles d'inférences

Pour les trois entrées, nous avons caractérisé 13 règles d'inférence:

- Si (A_i-A_j est *Faible*) alors (B_u-B_v est *Faible*)
- Si (A_i-A_j est Moyen) et (A_i-B_u est Faible) alors (B_u-B_v est Faible)
- Si (A_i-A_j est Moyen) et (A_j-B_v est Faible) alors (B_u-B_v est Faible)
- Si (A_i-A_j est Moyen) et (A_i-B_u est Moyen) et (A_j-B_v est Moyen) alors (B_u-B_v est Moyen)

- Si (A_i-A_j est *Moyen*) et (A_i-B_u est *Fort*) et (A_j-B_v est *Moyen*) alors (B_u-B_v est *Fort*)
- Si (A_i-A_j est *Moyen*) et (A_i-B_u est *Moyen*) et (A_j-B_v est *Fort*) alors (B_u-B_v est *Fort*)
- Si (A_i-A_j est *Moyen*) et (A_i-B_u est *Fort*) et (A_j-B_v est *Fort*) alors (B_u-B_v est *Fort*)
- Si (A_i-A_j est Fort) et (A_i-B_u est Faible) et (A_j-B_v est Faible) alors (B_u-B_v est Faible)
- Si (A_i-A_j est *Fort*) et (A_i-B_u est *Faible*) et (A_j-B_v n'est pas *Faible*) alors (B_u-B_v est *Moyen*)
- Si (A_i-A_j est *Fort*) et (A_i-B_u n'est pas *Faible*) et (A_j-B_v est *Faible*) alors (B_u-B_v est *Moyen*)
- Si (A_i-A_j est *Fort*) et (A_i-B_u est *Moyen*) et (A_j-B_v n'est pas *Faible*) alors (B_u-B_v est *Fort*)
- Si (A_i-A_j est *Fort*) et (A_i-B_u n'est pas *Faible*) et (A_j-B_v est *Moyen*) alors (B_u-B_v est *Fort*)
- Si (A_i-A_j est *Fort*) et (A_i-B_u est *Fort*) et (A_j-B_v est *Fort*) alors (B_u-B_v est *Fort*)

Ces 13 règles d'inférence résument les 27 combinaisons possibles des entrées pour construire la sortie. Nous présentons dans le tableau IV-3 toutes les relations entre les entrées et la sortie.

	A_i-A_j	**A_i-B_u**	**A_j-B_v**	**B_u-B_v**
1	Faible	Faible	Faible	Faible
2	Faible	Faible	Moyen	Faible
3	Faible	Faible	Fort	Faible
4	Faible	Moyen	Faible	Faible
5	Faible	Moyen	Moyen	Faible
6	Faible	Moyen	Fort	Faible
7	Faible	Fort	Faible	Faible
8	Faible	Fort	Moyen	Faible
9	Faible	Fort	Fort	Faible
10	Moyen	Faible	Faible	Faible
11	Moyen	Faible	Moyen	Faible
12	Moyen	Faible	Fort	Faible
13	Moyen	Moyen	Faible	Faible
14	Moyen	Moyen	Moyen	Moyen
15	Moyen	Moyen	Fort	Fort
16	Moyen	Fort	Faible	Faible
17	Moyen	Fort	Moyen	Fort
18	Moyen	Fort	Fort	Fort
19	Fort	Faible	Faible	Faible
20	Fort	Faible	Moyen	Moyen
21	Fort	Faible	Fort	Moyen
22	Fort	Moyen	Faible	Moyen
23	Fort	Moyen	Moyen	Fort
24	Fort	Moyen	Fort	Fort
25	Fort	Fort	Faible	Moyen
26	Fort	Fort	Moyen	Fort
27	Fort	Fort	Fort	Fort

Tableau IV-3. Expression de la sortie en fonction des entrées

Les règles d'inférences que nous venons de présenter permettent ainsi de lier l'intensité d'interaction entre deux éléments B_u-B_v pour tout couple d'éléments A_i-A_j.

6.2.4. Construction de la DSM finale

Le processus flou nous permet de construire une DSM B pour chaque couple d'élément (A_i, A_j). Cependant, la première règle qui formalise la construction de la DSM finale stipule qu'il est possible de coupler de deux manières différentes deux éléments de A et de B. C'est pourquoi nous faisons le choix d'exprimer la valeur finale de $DSM_{(A_i,A_j)}(B_u, B_v)$ comme suit :

$$DSM_{(A_i,A_j)}(B_u, B_v) = \frac{DSM(A_i \to B_u, A_j \to B_v) + DSM(A_i \to B_v, A_j \to B_u)}{2} \quad \textbf{Eq.IV-5}$$

Avec :

$DSM_{(A_i,A_j)}(B_u, B_v)$	*la valeur de l'interaction entre* B_u-B_v *en partant de* A_i-A_j
$DSM(A_i \to B_u, A_j \to B_v)$	*la valeur de l'interaction entre* B_u-B_v *avec dans la matrice d'incidence* A_i *est couplé à* B_u *et* A_j *est couplé à* B_v
$DSM(A_i \to B_v, A_j \to B_u)$	*la valeur de l'interaction entre* B_u-B_v *avec dans la matrice d'incidence* A_i *est couplé à* B_v *et* A_j *est couplé à* B_u

Le choix de faire la moyenne est justifié par le choix de la méthode globale d'agrégation qui est aussi la moyenne.

Ainsi la valeur finale de l'interaction entre B_u-B_v est telle que représentée par l'équation Eq.IV-6. On remarquera qu'on divise par le nombre de couples (A_i,A_j) considérés.

$$DSM(B_u, B_v) = \frac{\sum_i \sum_j DSM_{(A_i,A_j)}(B_u, B_v)}{taille(DSMA)^2} \quad \textbf{Eq.IV-6}$$

6.2.5. Filtrage de la DSM résultante

Avec le processus flou proposé pour la construction des DSM à partir d'une seule matrice d'incidence, nous avons identifié le biais lié à l'impossibilité d'obtenir des valeurs nulles. Le principe du traitement étant sensiblement le même dans le processus actuel, on retrouve alors le même biais. Pour préparer la DSM résultante, nous utilisons le même principe de filtrage que celui présenté dans le paragraphe 4.3.5.

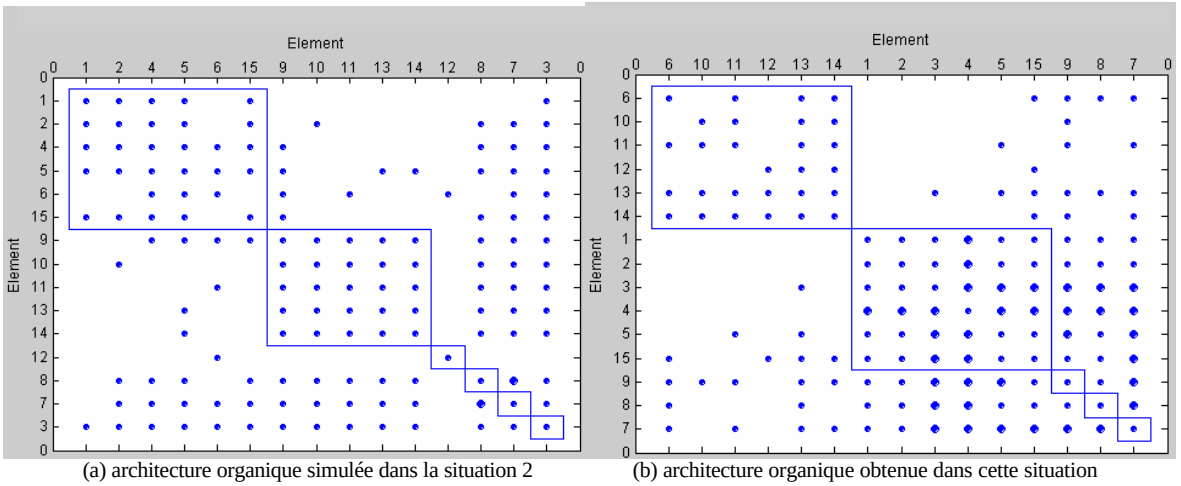
6.3. Application à l'architecture organique du produit

Nous allons reprendre l'exemple de la construction de l'architecture du produit. Nous proposons de construire l'architecture organique du moteur connaissant l'architecture du domaine des fonctions systèmes et la matrice d'incidence FS-COMP.

La matrice d'incidence est une donnée que nous avons déjà utilisée dans la deuxième situation de construction. Nous allons alors utiliser comme donnée la DSM FS construite directement par les acteurs du projet (Figure IV-8)

6.3.1. Construction de l'architecture organique

Dans ce paragraphe, nous analysons l'architecture organique (figure IV-38 (b)) obtenue à partir de la DSM FS et de la MI FS-COMP en la comparant à l'architecture obtenue avec la DSM COMP (figure IV-38 (a)) construite dans la situation 2 (directement à partir de la MI FS-COMP).



(a) architecture organique simulée dans la situation 2 (b) architecture organique obtenue dans cette situation

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EGR	Injection	Culasse	Admis.	Echapp.	Distrib.	Attelage mobile	Carter	Lubrif.	Entraîn. Access.	Entraîn. Synchro.	Circuit de vide	Circuit calopor.	GES.	CC

Figure IV-38. Comparaison des architectures organiques des situations 2 et 4

Nous comparons les deux architectures de la figure IV-38 comme suit :

- Les deux architectures sont composées de deux modules et trois éléments intégrateurs.
- Les modules sont très proches par leur composition, de même pour les éléments intégrateurs.
- Les éléments en commun dans le premier module sont : l'EGR, l'Injection, l'échappement et le Contrôle-Capteurs. Tous ces constituants appartiennent à la partie haute du moteur. Dans la DSM issue uniquement de la matrice d'incidence, nous trouvons en plus la distribution. L'algorithme a maintenant placé la Culasse, dans le premier module de la nouvelle DSM. Cet élément a été identifié dans la première comme étant intégrateur. La réalité concernant cet élément d'après les acteurs du projet est que ce constituant fait partie physiquement de la partie haute du moteur. Cependant, il a aussi un rôle intégrateur à la fois interne au module car les autres éléments se fixent sur lui et externe en complément du carter. Mais c'est le carter qui réalise l'intégration de la majorité des composants du moteur.
- Le deuxième module est composé des éléments invariants suivants : les deux entraînements et le circuit caloporteur. La lubrification qui a été identifiée comme faisant partie de ce module est devenue intégratrice tout en conservant un grand nombre

d'interactions avec ce même module. Nous retrouvons dans la nouvelle DSM le circuit de vide qui reprend une place attendue, tout en étant faiblement couplé aux autres éléments de ce module.

- L'attelage mobile et le carter sont les intégrateurs invariants entre les deux architectures. Ces deux éléments avec la contribution de la culasse constituent le squelette du moteur. En plus, ils sont couplés avec un grand nombre d'éléments. Le troisième élément intégrateur est la lubrification.

7. Synthèse

Après l'amélioration d'un algorithme de clustering au chapitre 3, nous avons présenté une deuxième contribution, portant sur une méthode de développement des architectures d'un système. Cette méthode est fondée sur l'identification de différentes situations de conception (cas d'utilisation d'un outil).

Les quatre situations inventoriées ont été accompagnées chacune par une démarche de construction des architectures. Ces démarches exploitent le formalisme matriciel des matrices d'incidence et des DSM en recourant à des opérations mathématiques et à un traitement flou quand il était nécessaire.

Le choix du traitement flou comme méthode de construction des DSM dans les deuxième et quatrième situations est cohérent avec le caractère subjectif des données manipulées, construites par les acteurs du projet.

La mise en œuvre de la méthode de construction sur l'exemple d'un moteur diesel a montré son utilisation possible et a permis une première validation. Rappelons à cet effet que l'objectif principal de cette méthode est d'aider les architectes systèmes à appréhender la complexité des systèmes dits complexes en leur permettant de simuler et d'analyser les architectures des domaines du produit. On peut imaginer qu'à la suite de cette étape l'architecte système puisse être en mesure d'adopter les architectures proposées et ainsi de considérer les modules comme des boîtes noires dont il figera les interfaces externes en arbitrant avec les éléments intégrateurs.

On peut considérer aussi qu'à la suite de l'application de cette méthode, l'architecte puisse identifier des incohérences en comparant les architectures de deux domaines couplés. Il voudrait dans ce cas apporter des modifications à l'un de ces domaines et simuler des architectures cohérentes pour chacun de ces domaines.

C'est en réponse à cette dernière problématique que nous exposons dans le chapitre suivant une méthode de coévolution des architectures en l'illustrant sur la coévolution des architectures du produit et de l'organisation du projet avec comme moteur de l'évolution, l'exploration des sources d'incertitudes.

CHAPITRE V

CHAPITRE V

VERS LA COEVOLUTION DES ARCHITECTURES DES DOMAINES COUPLES

Un projet de conception d'une famille de produits ou d'un produit complexe comporte, lors des phases préliminaires, des activités d'architecture critiques pour le coût et la qualité du produit, car elles génèrent des décisions qui vont fortement orienter ou contraindre les choix ultérieurs en conception détaillée. Lors de ces phases, des itérations rapides entre différents métiers seront réalisées pour vérifier la faisabilité de choix d'architecture et prendre en compte des évolutions dans l'architecture du produit. Ultérieurement, lors de projets de reconception du produit permettant de prolonger sa vie commerciale, des activités semblables devront être menées.

Dans le cadre du pilotage organisationnel et en considérant la modélisation de l'architecture globale du projet, il est évident que la conception des architectures passe par la propagation des modifications et des changements entre tous les domaines du projet. Ainsi, nous pensons que toute modification impactant un domaine a d'abord des répercussions sur son architecture et à cause du couplage entre les domaines, la propagation de ces modifications a pour conséquence de faire évoluer les caractéristiques des autres domaines et par la même leurs architectures.

Plusieurs travaux de recherche se sont intéressés, à l'évolution individuelle des architectures de chacun des domaines du projet, on peut citer [Balachandra, 2002 ; Clarkson et al., 2004 ; Chen et Liu, 2005 ; Keller et al., 2005 ; Avak, 2006] pour la modélisation de l'évolution de l'architecture du produit, Galbraith [1977; 1994] pour l'évolution de l'architecture de l'équipe de conception. Cependant, nous n'avons pas pu référencer des travaux modélisant à la fois la coévolution de ces deux domaines du projet.

Dans ce chapitre, nous faisons l'hypothèse que pour toute fonction ou composant du produit et ce, à n'importe quel niveau de décomposition, il existe une tâche de conception qui porte sur la définition de cet élément du produit. Les domaines du produit et des processus sont alors bijectifs ce qui aboutit à l'obtention des mêmes DSM Produit et Processus. Le choix de cette hypothèse est lié au traitement différent qu'aurait nécessité l'analyse de la DSM

Processus (partionnement) et l'information qu'elle contient (enchaînement des tâches) si on avait opté pour l'utilisation d'une DSM Processus temporelle. Cette DSM sort du cadre de notre travail et de son objectif qui est l'identification des architectures sur des DSM statiques.

Dans la suite de ce travail, nous ne modéliserons pas dans sa globalité l'organisation du projet. Nous nous limiterons à la modélisation du domaine organique de cette organisation par l'utilisation d'une DSM Acteurs. La DSM Acteurs est alors couplée au produit par une matrice d'incidence MI-Produit-Acteurs.

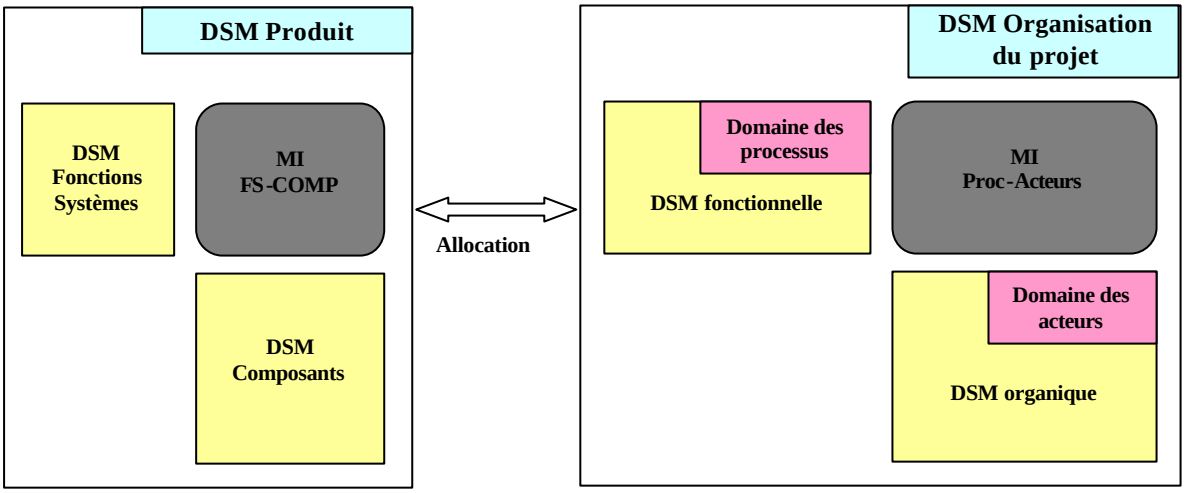


Figure V- 1. Modélisation des domaines du produit et de l'organisation du projet

Ce chapitre s'articule de la manière suivante : d'abord nous présenterons le principe de la méthode, ensuite nous détaillerons une par une les étapes qui la compose. Nous proposerons une typologie des incertitudes qui nous permet de modéliser les changements introduits, ensuite nous formaliserons la méthode de coévolution des architectures en nous basant sur le traitement flou, l'outil DSM et l'algorithme de clustering. Enfin, cette méthode de coévolution sera appliquée à un exemple industriel portant sur l'évolution d'une Boîte de Vitesse (BV) manuelle vers une BV robotisée.

1. La méthode de coévolution des architectures

Dans le chapitre précédent, nous avons présenté plusieurs méthodes pour la construction des architectures des domaines du produit. Nous avons montré l'existence de couplages entre ces domaines. Cependant, un projet de conception a aussi une dynamique temporelle et les architectures des domaines qui le composent peuvent évoluer au gré des levées d'incertitudes et des modifications demandées. Ainsi, vu que tous les domaines sont liés, toute modification qui touche un domaine influence tous les autres domaines du projet.

Nous entendons développer et mettre en œuvre une méthode de propagation entre les architectures de deux domaines A et B couplés à travers une matrice d'incidence A-B. Cette

propagation est modélisée comme l'évolution d'une situation initiale (caractérisée par DSM A_0, DSM B_0 et la matrice d'incidence MI A_0-B_0) vers une situation finale (caractérisée par DSM A_f, DSM B_f et la matrice d'incidence MI A_f-B_f). La transition d'une situation initiale stable vers une situation finale stable nécessite le passage par plusieurs étapes intermédiaires où les incertitudes sont introduites progressivement puis propagées.

La méthode de propagation que nous proposons est structurée comme suit :

1. Modéliser la situation initiale :
    - Construire les DSM A_0 et DSM B_0 et la matrice d'incidence MI A_0-B_0
    - Vérifier et réaliser la cohérence de la situation initiale
2. Explorer les incertitudes introduites par l'évolution de la situation initiale : construction de DSM A_i et DSM B_i et MI A_i-B_i
3. Simuler à partir de la situation intermédiaire les architectures finales avec la méthode de coévolution.

2. Principe de cohérence entre les architectures couplées

Si les domaines du projet sont couplés alors leurs architectures le sont aussi. L'existence de ce couplage sous-entend d'un côté, que toute évolution dans l'architecture d'un domaine doit être répercutée sur les architectures des autres domaines et d'un autre côté, que l'architecture d'un domaine est le fruit de la propagation de toutes les contraintes d'architectures des autres domaines.

Sur la base de cette dernière constatation, nous faisons l'hypothèse que si on modélise l'architecture de tous les domaines du projet à un instant donné et si ces architectures n'évoluent pas, alors ces architectures sont en équilibre et elles sont cohérentes entre elles. Autrement dit, lorsqu'aucune modification n'est introduite dans le projet, les architectures sont stables et donc optimales vis-à-vis des propagations des contraintes entre les domaines.

Nous considérons que les méthodes développées dans le chapitre précédent concourent à l'obtention d'architectures cohérentes et ce, parce qu'elles reposent sur l'utilisation des matrices d'incidence pour identifier les DSM et donc les architectures.

La méthode de coévolution que nous introduisons dans ce chapitre n'a de sens que si les situations initiales et finales du problème sont cohérentes et optimales. Si par construction la méthode que nous proposons garantit la cohérence des architectures finales, nous devons proposer un test de cohérence sur les architectures initiales.

2.1. Modélisation et Validation de la situation initiale

Pour valider la situation initiale du point de vue cohérence des architectures, nous proposons une démarche en deux points :

2.1.1. Test de cohérence des architectures initiales

Pour tester la cohérence des architectures initiales, nous proposons d'utiliser le rôle pivot des matrices d'incidence pour lier les architectures de deux domaines couplés. Puisque nous avons affirmé que les architectures obtenues à partir d'une matrice d'incidence sont cohérentes, nous proposons alors aux acteurs du projet de comparer les architectures construites aux architectures simulées à partir de la matrice d'incidence. Nous avons opté pour la non-automatisation de cette étape de comparaison qui permettra aux acteurs de projet de confronter la réalité du projet aux architectures obtenues par simulation.

2.1.2. Modification de la situation initiale

Si le test révèle un écart conséquent entre les architectures proposées et les architectures simulées ou si les acteurs préfèrent les architectures simulées, nous proposons deux approches pour proposer des architectures initiales cohérentes. Ces deux approches reposent sur les méthodes proposées dans le chapitre précédent.

- *Première approche* : elle s'applique quand les DSM et les architectures identifiées ne reflètent pas l'existence d'une politique de l'entreprise pour la propagation des contraintes d'architectures entre les domaines du projet. Dans ce cas, il n'est pas possible de considérer la situation initiale comme étant une situation optimale. La seule donnée de confiance qui reste est la matrice d'incidence. Nous appliquons alors la méthode de construction de deux DSM en partant d'une matrice d'incidence, méthode exposée dans la partie 4 du chapitre 4. Si les architectures obtenues sont validées par les acteurs du projet, ces architectures peuvent servir comme modèle de la situation initiale.
- *Deuxième approche* : elle s'applique quand l'architecture de l'une des DSM décrivant la situation initiale est validée par les acteurs du projet. Dans ce cas, nous pouvons proposer une architecture cohérente de l'autre domaine en utilisant la méthode de propagation introduite dans la partie 6 du chapitre IV. Cette méthode de propagation utilise une DSM et une MI pour construire une nouvelle DSM. Les architectures obtenues à l'issue de cette approche, si elles sont validées par les acteurs du projet, offre des architectures cohérentes pour décrire la situation initiale.

Si dans la démarche de validation des architectures initiales, nous utilisons une des deux approches de restructuration présentées ci-dessus, cela sous-entend que les architectures observées ne sont pas cohérentes et que la situation initiale aurait pu être améliorée avant de subir les perturbations qui vont la faire évoluer.

3. L'exploration des incertitudes

Notre démarche de simulation de l'évolution des architectures couplées du projet est en adéquation avec les objectifs du pilotage de projet de conception en matière d'adaptation aux incertitudes qui peuvent intervenir dans le projet. Ce positionnement fait que nous nous sommes inspirés d'une part, des travaux de Ward et Chapman [Chapman et Ward, 1997] [Ward et Chapman, 2003] en matière de management de l'incertitude pour définir un cadre au mangement de l'incertitude et d'autre part, des travaux de Loch, Pich et De Meyer [Loch et al., 2000 ; Pich et al., 2002] pour proposer une typologie de l'incertitude.

3.1. Les typologies références

3.1.1. Le management de l'incertitude selon Chapman et Ward

Chapman et Ward [2003] présentent le management de l'incertitude comme une approche qui a été initialement introduit pour donner un contrepoids à la gestion des risques en introduisant la possibilité de considérer les aléas comme une opportunité. Cependant, ces chercheurs affirment que le management de l'incertitude a dépassé le cadre de cette opposition entre les menaces et les opportunités pour s'intéresser aussi aux sources des incertitudes, avant d'établir un plan d'action, pour les gérer et les classer en désirables ou non.

Dans ce cadre, l'incertitude définie par Chapman et Ward (et telle que nous l'utiliserons) s'apparente à une absence (ou un manque) d'informations autour de l'entreprise et plus précisément, dans les outils, modèles et données qui caractérisent l'entreprise et son fonctionnement. Toutefois, la typologie que proposent ces auteurs est beaucoup plus managériale et moins formelle que celle que nous visons, elle se résume comme suit :

- Incertitude sur la conception et la logistique : la nature des livrables en conception et des processus qui les créent est un aspect fondamental de l'incertitude dans le projet. En principe, une grande part de cette incertitude est éliminée dans les stades amonts du cycle de vie du produit en spécifiant ce qui doit être fait, quand, comment, par qui et à quel prix. En pratique, une part appréciable de l'incertitude du projet persiste tout au long du cycle de vie du produit.

- Incertitude sur les relations fondamentales : la multiplication des acteurs humains et des types d'entités organisationnelles (internes ou externes à l'entreprise) impliqués dans le projet forment une réelle source d'incertitude dans l'entreprise. Les relations entre ces différentes parties peuvent être complexes et ne pas utiliser des canaux formels. L'implication de plusieurs parties dans un projet introduit de l'incertitude (par ambigüité) autour des rôles et des responsabilités ainsi que sur les interactions formelles entre ces parties.

- Incertitude autour des objectifs et des priorités : avoir comme objectif l'amélioration de la performance du projet présuppose d'avoir une vision claire des objectifs du projet et des concessions à faire entre les objectifs et les contraintes. Essayer de gérer un projet quand cette vision manque est comme essayer de bâtir une tour sur du sable mou.
- Variabilité : la caractérisation des paramètres de projet est une source réelle d'incertitude. Par exemple, il est possible d'ignorer quelles ressources et quelle durée sont nécessaires pour l'accomplissement d'une tâche. Cette incertitude est souvent liée à un manque d'information ou de l'imprécision, plutôt que liée à une méconnaissance des risques et des aléas qui peuvent toucher le projet.
- Incertitude autour de la base des estimations : un champ important des incertitudes sont les bases des estimations produites par les différentes parties du projet. Par exemple, il est souvent nécessaire de se baser sur des évaluations subjectives et des probabilités en absence de données statistiques appropriées qui permettent de déterminer "objectivement" les probabilités. L'incertitude au sujet de la base des évaluations peut dépendre de plusieurs facteurs méconnus ou inconnus : qui les a produites, sous quelles formes, pourquoi, comment et quand ont-elles été produites, à l'aide de quelles ressources ou sur la base de quelles expériences ?
- Incertitude associée à la nature conditionnelle des estimations. Les estimations produites dans le cadre d'un projet sont toutes contraintes par des hypothèses et elles ne sont vraies ou justes que dans le cadre de ces hypothèses. Une source particulièrement importante d'incertitudes sur les estimations est liée à la non vérification et la non validation des hypothèses.

3.1.2. La typologie d'incertitude selon Loch, Pich et De Meyer

Un projet est communément vu comme l'agencement d'activités parallèles et séquentielles qui permettent de créer de la valeur [Morris et al., 1987], [Meridith et al., 1995]. Loch et ses coauteurs [Loch et al., 2000] conceptualisent le projet non comme un ensemble de tâches à réaliser, mais comme un ensemble de facteurs qui influent sur la création de valeur.

Ils définissent la valeur créée par Π, qui est une fonction de N paramètres w_i caractérisant l'entreprise et les projets qu'elle développe : $\Pi = \Pi(w_1,...,w_N)$.

Les auteurs affirment que la plus-value du projet ne peut être prédite avec certitude, puisqu'il faut caractériser un à un les paramètres w_i à qui on associe un domaine de définition D_i. Les sources d'incertitudes identifiés par les auteurs et liées à la caractérisation de Π sont au nombre de 5 : la complexité, la variabilité, les risques, l'ambigüité et le chaos.

3.1.2.1. La complexité

Si on considère que les paramètres w_i sont directement déterminés par les responsables de l'entreprise. Ces paramètres peuvent être interdépendants et l'influence de cette interdépendance sur Π peut échapper à ces responsables. Plus les couplages sont nombreux, plus la complexité du modèle est grande. Ainsi, l'incertitude par complexité est liée à ces couplages entre paramètres. S'il y a incertitude par complexité sur Π alors les responsables sont incapables de spécifier tous les couplages entre les paramètres w_i impliquées dans la caractérisation de Π.

3.1.2.2. La variabilité

Certains paramètres déterminant Π peuvent ne pas être accessibles individuellement un par un et ce par manque de moyens (coût important) ou par incapacité technique. Par contre, il est possible d'avoir accès à l'effet global de ces paramètres par une fonction de distribution F. les auteurs proposent alors de transformer Π en une fonction de distribution en fonction de F. L'incertitude par variabilité peut alors être estimée en calculant la variance de Π.

3.1.2.3. Les risques

Cette situation est caractérisée par l'existence de certains paramètres sous la forme de variables aléatoires. Les auteurs utilisent des mesures de probabilité pour chacun de ces paramètres et proposent d'estimer le risque introduit par ces variables en utilisant la variance de Π.

3.1.2.4. L'ambiguïté

L'équipe projet peut occulter (ou ne pas connaître) l'existence de certains paramètres. On considère $(w_{L+M+1},...,w_N)$ ces paramètres. Selon toute logique, l'équipe ignore l'impact de ces paramètres sur la valeur du projet, ce qui revient à leur donner une valeur par défaut dans le modèle réel.

En effet, si on considère que l'expression utilisée de Π est $\Pi = X - Y - Z^2$. Cependant l'expression réelle est $\Pi = X - W * Y - Z^2$. Dans ce cas, implicitement et en ignorant l'existence de W, les responsables du projet utilisent inconsciemment le modèle réel avec W toujours égale à 1. Alors que dans la réalité W peut prendre une autre valeur que 1.

Une estimation de l'influence de l'incertitude par ambigüité sur Π est réalisée en calculant l'amplitude des écarts sur les valeurs observées de Π.

Par définition même l'incertitude par ambigüité ne peut être anticipée puisqu'elle résulte de la non connaissance de l'existence de ces paramètres. Dans ce cas là, il est seulement possible de l'explorer au cours du projet en identifiant de nouveaux paramètres.

3.1.2.5. Le chaos

Supposons que les paramètres $(w_{L+M+1},\ldots,w_N)$ sont interdépendants par une fonction $h(w_{L+M+1},\ldots,w_N)=c$. Le projet est chaotique dans le sens où un changement minime dans un paramètre peut causer une complète réévaluation de tous les paramètres [Cohen et al., 1994]. Dans cette situation, il n'est pas possible d'estimer la valeur Π du projet.

3.1.2.6. Récapitulatif

Le tableau V-1 résume la typologie adoptée par Loch, Pich et De Meyer, on y fait référence aux outils de représentation utilisés dans l'entreprise tels que les graphes et les arbres de décision et les méthodes telles que PERT et GERT.

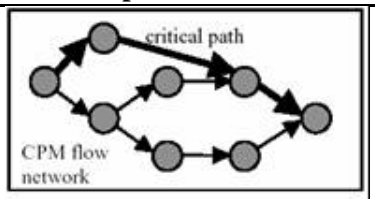
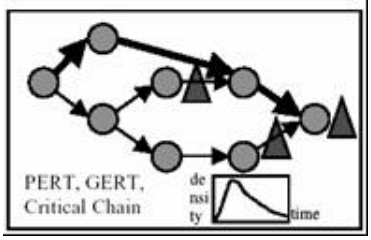
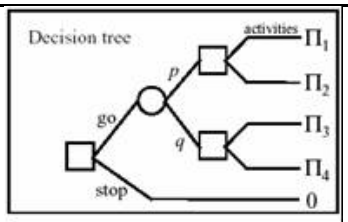
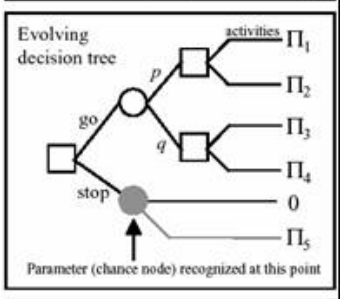
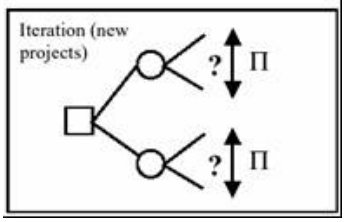
Représentation	Type	Structure
 	Complexité	$w=\begin{pmatrix} w_1 \\ \ldots \\ w_N \end{pmatrix}$ connu
	Variabilité	$(w_1,\ldots,w_L)$ ne sont pas individuellement connus : seulement leur influence collective peut être estimée par une fonction de distribution
	Risques	$(w_L,\ldots,w_{L+M})$ sont des variables aléatoires.
	Ambiguïté	L'équipe projet n'est pas consciente de l'existence de $(w_{L+M+1},\ldots,w_N)$. Le projet est basé sur des paramètres par défaut $(w^*_{L+M+1},\ldots,w^*_N)$
	Chaos	Les paramètres $(w_{L+M+1},\ldots,w_N)$ Sont interdépendant. Leur influence sur la valeur du projet ne peut pas être évaluée et leur influence ne peut être étudiée un par un.

Tableau V-1. Typologie des incertitudes selon Loch et al. [2000]

Cette typologie en cinq points se montre efficace pour une modélisation mathématique du projet basée sur les paramètres et attributs qui caractérisent tout élément impliqué dans le projet.

3.1.3. Analyse des typologies de référence

Il est facile de remarquer que les deux typologies que nous avons référencées reflètent deux niveaux de pilotage des projets en entreprise. La première typologie se situe à un niveau managérial et procédural tandis que la deuxième est beaucoup plus opérationnelle.

D'après la typologie de Chapman et Ward (Chapman et al. 2003), l'incertitude peut être issue de :

- La variabilité liée au choix de la valeur à attribuer aux paramètres ou informations traités ;
- L'ambiguïté liée au sens et à l'utilisation du paramètre ou de l'information.

La typologie de Loch, Pich et De Meyer [Loch et al., 2000 ; Pich et al., 2002] nous la résumons comme suit :

- L'incertitude par ambigüité caractérise l'utilisation d'un modèle qui ne prend pas en compte tous les paramètres ou toutes les entités. L'ambigüité dans son sens littéral provient alors des singularités qui peuvent avoir lieu sans qu'elles puissent être expliquées par le modèle tronqué.
- La complexité et le chaos sont liés à la difficulté de construire des modèles fidèles lorsque le nombre d'éléments ou le nombre d'interactions devient très grand.
- La variabilité et les risques sont liés à la variabilité des éléments qui caractérisent le modèle.

Dans notre cas, nous recherchons une typologie qui se situe entre les deux typologies de référence, une typologie qui puisse être utilisée dans les phases amont des projets de conception après que les décisions managériales aient été prises au niveau de l'entreprise et avant qu'on ait une connaissance approfondie de tous les paramètres caractérisant le projet.

3.2. Notre typologie de l'incertitude

Une typologie appropriée de l'incertitude peut orienter la démarche d'exploration des incertitudes qu'on désire appliquer. Cette typologie sera considérée comme satisfaisante si elle permet de couvrir toutes les sources d'incertitude.

L'interprétation que nous réalisons des deux typologies de référence nous amène à conclure :

- comparativement à la deuxième typologie, la première ne prend pas en compte la complexité comme source d'incertitude.

- la deuxième typologie peut être condensée en trois types d'incertitudes.

A la lumière de ces remarques, nous proposons la typologie suivante en 3 points [Harmel et al., 2006a ; 2006d] :

- L'incertitude par ambigüité : elle est liée, soit à la non identification d'un élément appartenant au système, soit à la prise en compte d'un élément qui ne doit pas appartenir au système (introduction de E sur la figure V-2).
- L'incertitude par complexité : elle est liée, soit à la non identification d'un couplage ou d'une interaction entre deux éléments du système (introduction des couplages B-A, A -E et E-C sur la figure V-2), soit à la prise en compte d'interactions qui ne doivent pas exister.
- L'incertitude par variabilité : elle est liée, soit à la non identification d'un attribut ou la non prise en compte de certaines valeurs pour caractériser un élément ou une interaction, soit à la prise en compte de valeur erronées ou impossibles (modification du domaine de définition de C sur la figure V-2).

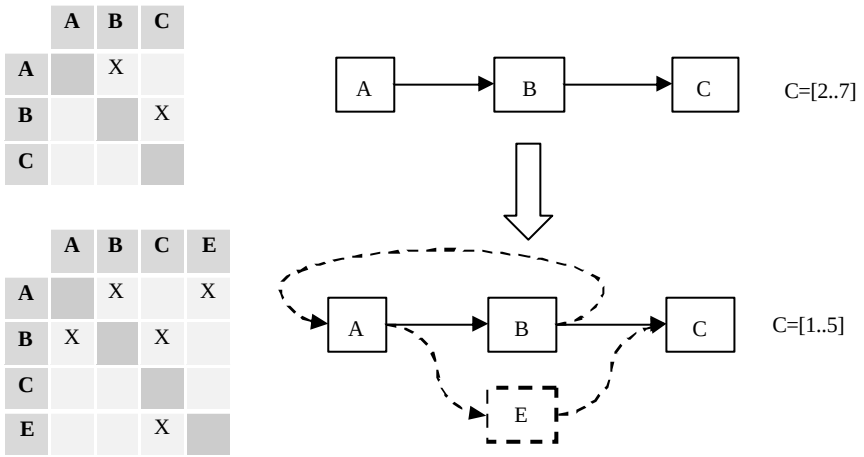


Figure V-2. Illustration de notre typologie d'incertitude

L'intérêt de cette typologie pour notre travail est accentué par le fait qu'une représentation sous forme de matrices est possible. Nous utiliserons cette typologie pour faire évoluer les matrices DSM.

3.3. La méthode d'exploration des incertitudes

Notre typologie des incertitudes énumère les classes d'incertitude dans un ordre qui reflète une hiérarchie entre elles. En effet, lexploration des sources d'incertitudes par ambigüité aboutit à l'identification de nouveaux éléments qui n'étaient pas pris en compte dans la représentation du système. Tout nouvel élément ne peut appartenir au système que si on identifie des couplages le liant aux autres éléments du système. Ainsi, la levée des incertitudes

par ambigüité fait appel à l'exploration des sources d'incertitudes par complexité. Cette deuxième étape peut aboutir aussi à l'identification de nouveaux couplages entre les anciens éléments du système. Pour finaliser la caractérisation du système, il est nécessaire d'identifier les nouveaux éléments et les nouveaux couplages et de fixer la valeur des attributs de chacun. Il s'agit de l'exploration des sources d'incertitudes par variabilité.

La méthode d'exploration (représentée sur la figure V-3) garantit l'exploration des incertitudes qui peuvent caractériser un système. Tout autre ordre dans l'exploration des sources d'incertitudes ne peut pas être efficace en raison de la filiation qui existe entre les trois types d'incertitudes.

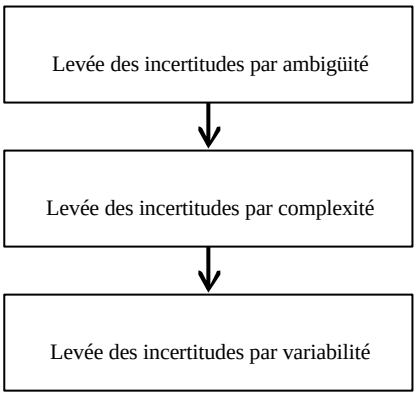


Figure V-3. Processus d'exploration des incertitudes

4. Méthode de coévolution des architectures

La méthode de coévolution des architectures débute après la levée des incertitudes et la construction du modèle de la situation intermédiaire.

La figure V-4 montre l'organigramme de la méthode globale de coévolution des architectures de deux domaines A et B.

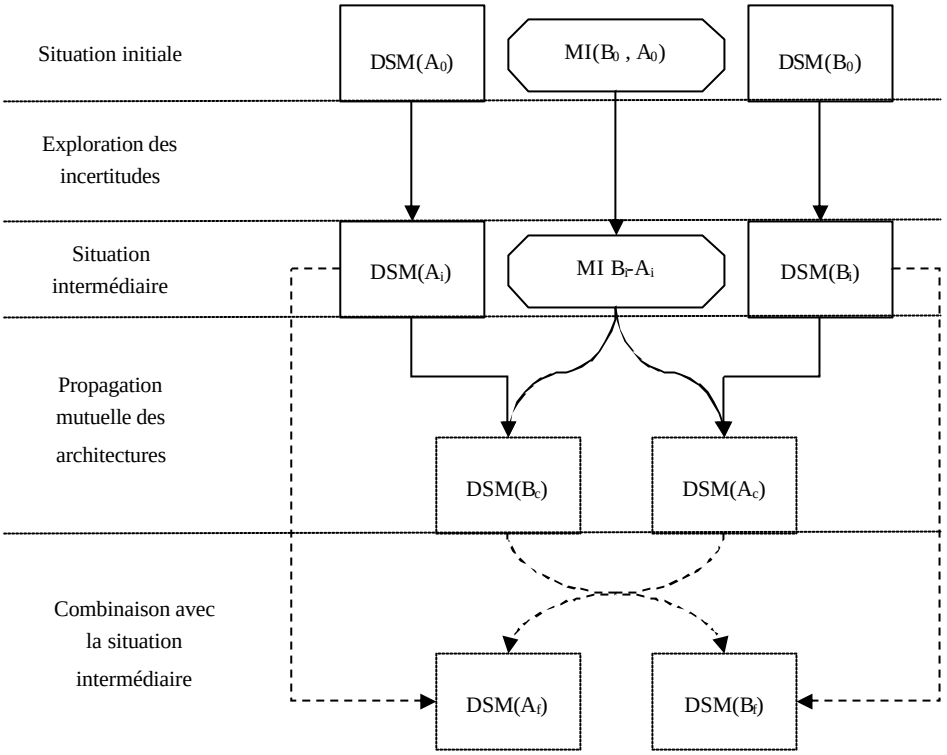


Figure V-4. Détail de la méthode de coévolution des architectures

Dans la partie suivante, nous allons expliciter les deux dernières étapes de l'organigramme.

4.1. Propagation mutuelle des architectures

La propagation mutuelle des architectures repose sur la méthode de propagation d'une architecture d'un domaine donné vers un autre. Dans la méthode de coévolution, nous allons réaliser une double propagation. Ainsi nous partons de $DSMA_i$ (respectivement, $DSMB_i$) pour obtenir $DSMB_c$ (respectivement, $DSMA_c$).

La méthode de construction de l'architecture de B en partant de l'architecture de A et de la MI A-B repose sur un traitement flou présenté au chapitre précédent (Situation IV).

La figure V-5 montre le détail du traitement flou réalisé pour obtenir les $DSMA_c$ et $DSMB_c$. Nous déroulons l'exemple de propagation de deux interactions $\left(A_n^i, A_m^i\right)$ de DSM A_i et $\left(B_k^i, B_j^i\right)$ de DSM B_i.

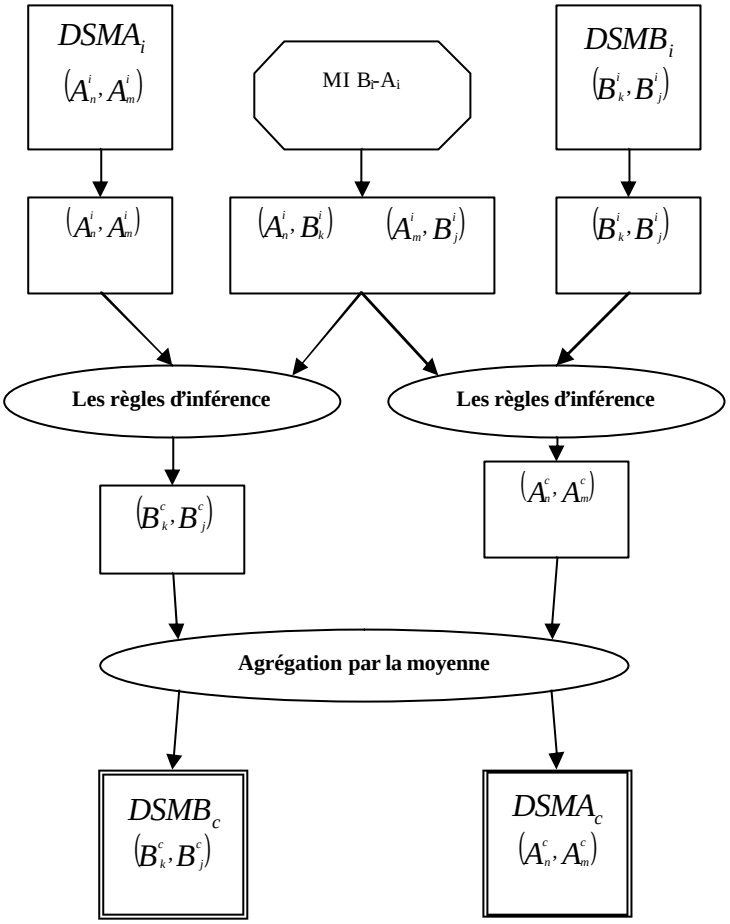


Figure V-5. Détail de la méthode de propagation mutuelle des architectures

On remarque à travers la figure ci-dessus que les méthodes de construction des $DSMA_c$ et $DSMB_c$ sont symétriques et passent par les mêmes étapes.

Pour justifier la nécessité d'utiliser cette méthode de coévolution lors d'une modification de la situation de conception, nous insistons sur les points suivants :

- Après exploration des sources d'incertitudes, les deux DSM et la MI obtenues ne sont plus forcément cohérentes, étant donné qu'elles ont été construites manuellement et d'une manière indépendante avec la possibilité que cela soit fait par deux équipes différentes.
- La méthode de coévolution permet de simuler l'architecture des domaines connaissant leur dual. Ainsi $DSMA_c$ est l'image de l'architecture de $DSMB_i$. Nous obtenons ainsi des architectures cohérentes croisées, d'un côté $DSMA_c$ avec $DSMB_i$ et de l'autre côté $DSMB_c$ avec $DSMA_i$.
- La méthode que nous proposons n'impose aucune hiérarchie entre les domaines. Donc, on obtient, à la fin de l'opération de coévolution, des architectures qui sont cohérentes avec la situation intermédiaire mais croisées.

Cette dernière remarque montre la nécessité d'introduire une opération de construction des architectures finales qui permet de lier, pour un même domaine, l'architecture intermédiaire et l'architecture coévoluée.

4.2. Construction des architectures finales

Pour l'utilisateur qui le juge nécessaire, nous offrons la possibilité de construire les architectures finales de chaque domaine en les combinant aux architectures intermédiaires (figure V-6).

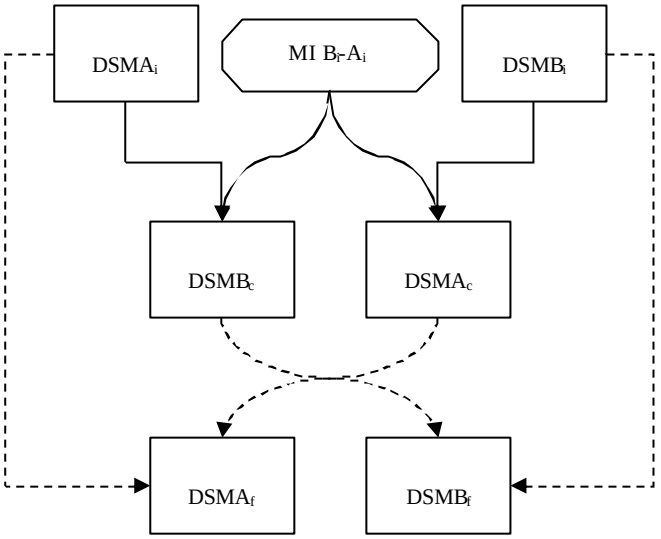


Figure V-6. Méthode de construction des architectures finales

La méthode de construction des architectures finales que nous proposons s'appuie sur une combinaison linéaire dont les paramètres a et ß sont à fixer par l'utilisateur (Eq.V-1).

$$DSMA_f = a\,DSMA_c + b\,DSMA_i \qquad a + b = 1 \qquad \textbf{Eq.V-1}$$

Il est conseillé de recourir à la normalisation des DSM coévoluées pour qu'elles soient comparables aux DSM intermédiaires construites manuellement.

Si les domaines sont réellement symétriques et qu'il n'y a pas de hiérarchie entre eux, il peut être souhaitable d'utiliser la moyenne pour construire les architectures finales.

5. Application de la méthode de coévolution des architectures

5.1. Présentation de la situation de conception

Pour mettre en œuvre la méthode de coévolution que nous proposons, nous avons travaillé sur la situation de conception d'une Boîte de Vitesse (BV) [Harmel et al., 2006c].

La boîte de vitesse en question est une BV manuelle. Le constructeur automobile a décidé de lancer un projet de reconception sur cette BV pour proposer une BV robotisée.

Une BV robotisée est une boîte de vitesse manuelle qui agit comme une BV automatique à travers l'incorporation d'un actionneur réalisant automatiquement, selon un schéma de commande prédéfini, les changements de vitesse. Ce projet est réellement un projet de reconception étant donné que le projet n'est pas parti de la feuille blanche mais de la BV manuelle pour réaliser les adaptations nécessaires pour en faire une BV robotisée.

Dans notre approche de modélisation de l'évolution des architectures dans le projet de reconception de la BV manuelle, nous avons opté pour la modélisation de la coévolution des architectures du produit et de l'équipe de conception.

La situation initiale est décrite par les DSM P_0 et DSM A_0 en figure V-7 et V-8 et par la matrice d'incidence MI P_0-A_0 représentée sur la figure V-9. Ces matrices ont été remplies et validées, à partir d'entretiens avec les différents acteurs du projet. Le tableau V-2 résume les principales abréviations utilisées dans les matrices que nous avons construites.

FS VOL	Fonction Système Volumes fonctionnels
FS TPU	Fonction Système Transmission de Puissance
FS ELU	Fonction Système reprise d'Effort et Lubrification
FS COU	Fonction Système Couplage
FS COM	Fonction Système Commutation
CDI	Commande Interne
SYN	Synchroniseur
TRI	Tringlerie
PE	Pédale d'Embrayage
EMB	Embrayage
CIE	Commande Interne d'Embrayage
DIFF	Différentiel
MEI	Mécanique Interne
CART	Carter

PAM	Chef de projet
PMIV	Responsable Intégration et validation
AF	Architecte fonctionnel
CdP	Chargé de projet de conception

Tableau V-2. Liste des éléments utilisées et leurs abréviations

	FS VOL	FS TPU	FS ELU	FS COU	FS COM	CDI	SYN	TRI	PE	EMB	CIE	DIFF	MEI	CART
FS VOL		7	8	7	8	6	7	8		6	4	7	8	8
FS TPU	7		7									8	9	
FS ELU	8	7		7	8		5			6			9	9
FS COU	7		7						8	9	7			
FS COM	8		8			7	7	8						
CDI	6				7		9	8						6
SYN	7		5		7	9							8	
TRI	8				8	8								8
PE				8							9			
EMB	6		6	9							8		6	
CIE	4			7					9	8				5
DIFF	7	8											7	8
MEI	8	9	9				8			6		7		9
CART	8		9			6		8			5	8	9	

Figure V-7. DSM P_0 : DSM Produit initiale

	PAM	PMIV	AF VOL	AF COM	CdP TRI	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP PE	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB
PAM		9	7	7	6	6	6	7	6	6	7	6	6	6	7	6	6
PMIV	9		7	7	8	8	8	7	8	8	7	8	8	8	7	8	8
AF VOL	7	7		7				7			6				8		
AF COM	7	7	7		9	9	9								7		
CdP TRI	6	8		9			9										
CdP SYN	6	8		9			9										
CdP CDI	6	8		9	9	9											
AF TPU	7	7	7						9	9							
CdP DIFF	6	8						9									
CdP MEI	6	8						9									
AF COU	7	7	6									8	9	9	7		
CdP PE	6	8									8			8			
CdP EMB	6	8									9			8			
CdP CIE	6	8									9	8	8				
AF ELU	7	7	8	7				7			7					8	8
CdP CART	6	8													8		9
CdP LUB	6	8													8	9	

Figure V-8. DSM A_0 : DSM Acteurs initiale

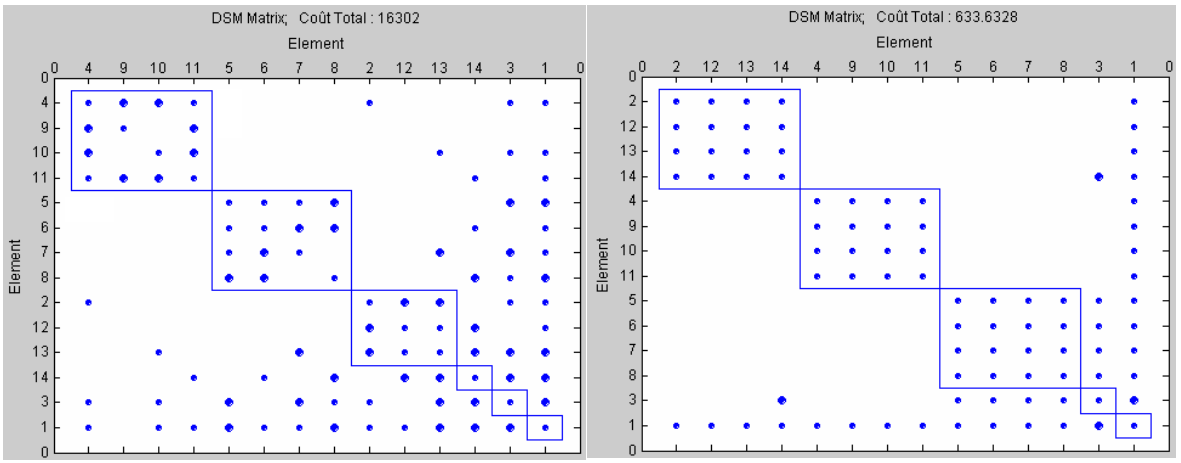
	PAM	PMIV	AF VOL	AF COM	CdP TRI	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP PE	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB
FS VOL	5	5	9	7				7			7				7		
FS TPU	5	5						9	8	8					5		
FS ELU	5	5	7	5											9	8	8
FS COU	5	5									9	7	7	7			
FS COM	5	5	7	9	7	7	7										
CDI	5	6		7	5	5	9										
SYN	5	6		7		9	5										
TRI	5	6		7	9		5										
PE	5	6									7	9		5			
EMB	5	6									7		9	5			
CIE	5	6									7	5	5	9			
DIFF	5	6						7	9	5							
MEI	5	6						7		9							8
CART	5	6								5					9	9	8

Figure V-9. MI P_0-A_0 : Matrice d'Incidence initiale

5.2. Vérification de la cohérence des architectures initiales

Pour vérifier la cohérence des architectures de la situation initiale, nous devons comparer les architectures initiales à celles obtenues à partir de la matrice d'incidence.

La figure V-10 montre l'architecture initiale du produit (a) et celle simulée à partir de la MI (b).



(a) architecture construite du produit (b) architecture simulée du produit

1	2	3	4	5	6	7	8	9	10	11	12	13	14
FS VOL	FS TPU	FS ELU	FS COU	FS COM	CDI	SYN	TRI	PE	EMB	CIE	DIFF	MEI	CART

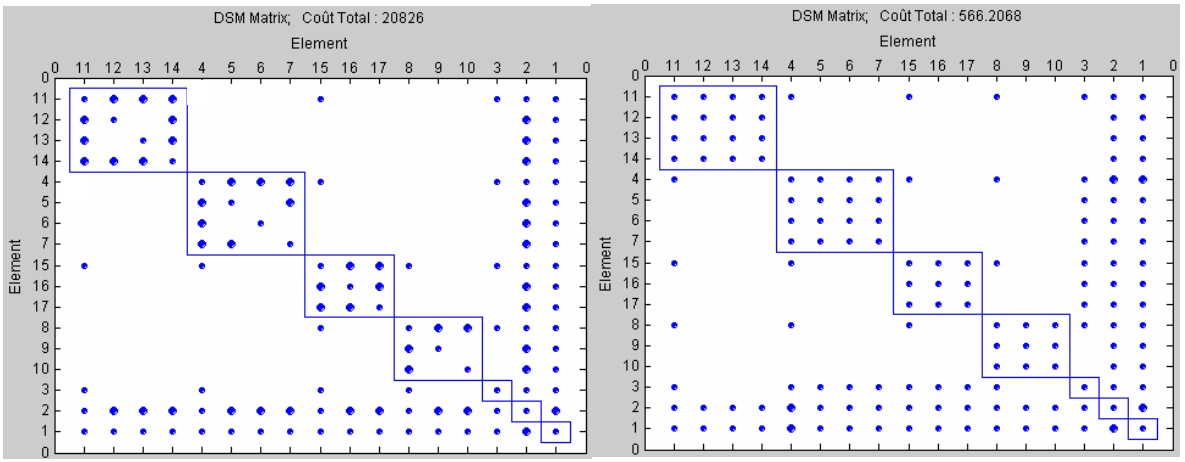
Figure V-10. Vérification de la cohérence de l'architecture initiale du produit

En comparant les deux architectures, nous remarquons :

- Les deux architectures comportent le même nombre de modules mais pas le même nombre d'éléments intégrateurs ;
- Après avoir construit la DSM P_0 avec les acteurs du projet, nous leur avons demandé d'identifier les éléments intégrateurs du produit. C'est ainsi que les deux FS Volumes fonctionnels (VOL) et reprise d'Effort et Lubrification (ELU) ont été désignées par les AF. Concernant les composants, le Carter ne faisait pas consensus quant à sont rôle d'intégrateur. Le clustering de la DSM P_0 identifie automatiquement les FS VOL et ELU. Quant au carter, nous l'avons désigné manuellement comme intégrateur ;
- Le clustering de la DSM P'_0 obtenue en partant de la matrice d'incidence montre que la FS VOL a été identifiée automatiquement comme étant intégrateur. Quant à la FS ELU, nous l'avons ajoutée manuellement. Nous avons choisi de ne pas caractériser le Carter comme intégrateur. Cette décision est appuyée par le fait qu'il s'intègre parfaitement dans un module bien spécifique. Ce dernier résultat nous a amené à simuler l'architecture initiale construite manuellement en libérant le Carter, on retrouve alors la même architecture que celle simulée.
- Les modules identifiés sont parfaitement les mêmes si on considère que le carter peut faire partie d'un module. Les modules correspondent aux principales fonctions non intégratrices de la BV. On trouve alors un module "transmission de puissance" avec FS TPU, DIFF et MEI, un module couplage avec FS COU, EMB, CIE et PE et un module commutation avec FS COM, TRI, SYN et CIE.

Pour conclure sur la cohérence de l'architecture perçue du produit par rapport à celle simulée, nous pensons (en accord avec les acteurs du projet) que les résultats sont satisfaisants et que le rôle du composant Carter ne remet pas en question l'architecture globale de la BV.

La figure V-11 montre l'architecture initiale de l'équipe de conception (a) et celle simulée à partir de la MI (b).



(a) architecture construite de l'équipe de conception ($DSMA_0$) (b) architecture simulée de l'équipe de conception ($DSMA_0$')

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
PAM	PMIV	AF VOL	AF COM	CdP TRI	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP PE	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB

Figure V-11. Vérification de la cohérence de l'architecture initiale des acteurs

Les deux architectures obtenues de l'équipe de conception suscitent les remarques suivantes :

- Les deux architectures sont identiques, les mêmes modules d'acteurs (acteurs collectifs) et les mêmes acteurs intégrateurs.
- Concernant la DSM A_0 construite manuellement, les entretiens ont fait ressortir que trois acteurs peuvent être considérés comme intégrateurs à savoir le PAM, le PMIV et AF VOL. Nous remarquons déjà que les acteurs intégrateurs ne peuvent pas être des concepteurs (CdP). Cela contredirait d'une part l'organisation hiérarchique (PAM, AF, CdP) et la logique d'organisation qui fait des CdP des spécialistes d'un domaine bien précis (dans notre cas, des composants). Le clustering de la DSM A_0, va dans le même sens que les attentes des acteurs, en identifiant le PAM et le PMIV comme étant intégrateurs. Quant à l'AF VOL, il a été ajouté manuellement, mais les interactions qu'il entretient avec les autres acteurs du projet révèlent son caractère intégrateur puisqu'il interagit avec toutes les équipes (modules) identifiées.
- Sur la base de la DSM A'_0, l'algorithme de clustering identifie automatiquement (IC=0.7) les éléments intégrateurs. Ce sont les mêmes que ceux identifiés dans DSM A_0.
- Les acteurs collectifs dans les deux architectures sont regroupés par FS avec des couplages entre acteurs qui sont réalisés exclusivement par les architectes fonctionnels.

En ce qui concerne les acteurs de conception, nous pouvons considérer que l'architecture perçue est cohérente avec celle qu'on pouvait obtenir à partir de la matrice d'incidence. Ceci nous permet aussi de conclure quant à la cohérence des architectures de la situation initiale dans sa globalité. Aucune correction n'est de ce fait nécessaire pour mettre en œuvre notre modèle de coévolution.

5.3. Exploration des sources d'incertitudes

L'exemple du projet de reconception mentionné plus haut va nous permettre d'illustrer les incertitudes qui peuvent être introduites par une innovation technologique dans un projet.

5.3.1. Exploration des incertitudes par ambigüité

L'objectif est ici d'identifier, par comparaison à la situation initiale, les nouveaux éléments qui doivent être introduits dans le système. Rappelons que nous nous sommes limités, dans la modélisation des domaines du projet, au couplage entre le produit et les acteurs de conception.

La levée des incertitudes par ambigüité dans la situation de conception de la BV robotisée conclut à :

- L'élimination des composants TRI et PE,
- L'élimination des acteurs CdP TRI et CdP PE,
- L'introduction d'un actionneur (ACT). Ce composant agit comme un automate en passant les vitesses et agissant sur l'embrayage selon des modes préprogrammés,
- L'introduction d'un acteur, Chargé de Projet de l'actionneur (CdP ACT).

Les figures V-12, V-13 et V-14 montrent les nouvelles listes d'éléments dans les DSM A_i, DSM P_i et MI A_i-P_i.

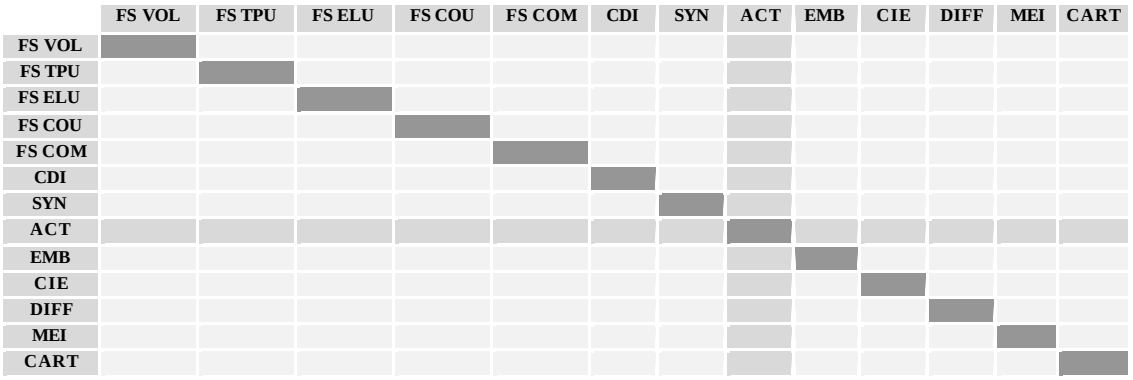


Figure V-12. Exploration des incertitudes par ambigüité dans DSM P_i

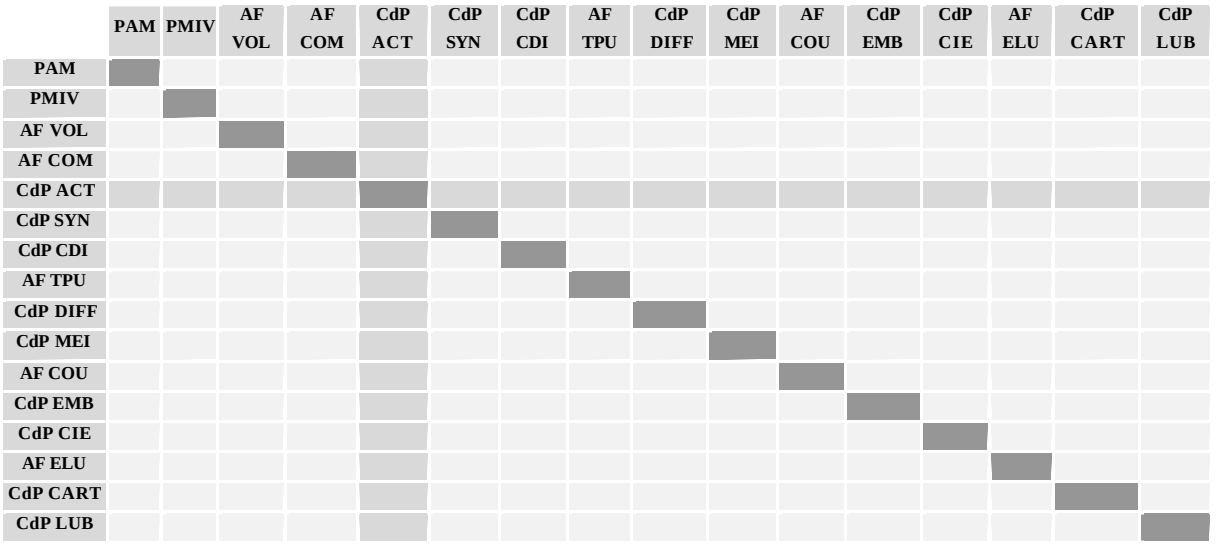


Figure V-13. Exploration des incertitudes par ambigüité dans DSM A_i

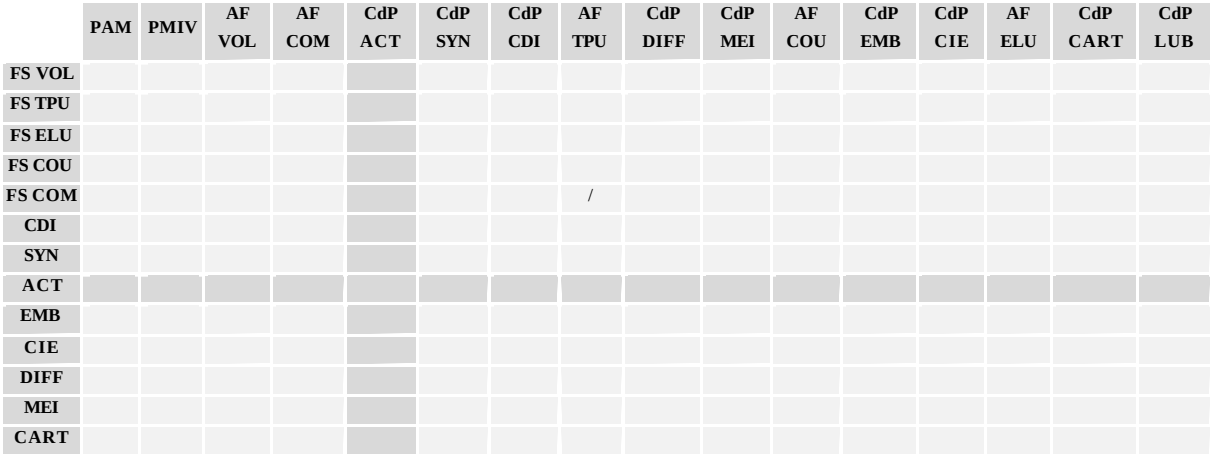


Figure V-14. Exploration des incertitudes par ambigüité dans MI P_i-A_i

5.3.2. Exploration des incertitudes par complexité

Après avoir identifié la nouvelle composition de l'architecture du produit et de l'équipe de conception, nous pouvons rechercher les couplages qui accompagnent l'introduction de ces nouveaux éléments. Pour identifier les couplages, nous utiliserons des croix. Les couplages mettant en œuvre les nouveaux éléments sont en gris foncé.

	FS VOL	FS TPU	FS ELU	FS COU	FS COM	CDI	SYN	ACT	EMB	CIE	DIFF	MEI	CART
FS VOL		X	X	X	X	X	X	X	X	X	X	X	X
FS TPU	X		X								X	X	
FS ELU	X	X		X	X		X	X	X			X	X
FS COU	X		X		X			X	X	X			
FS COM	X		X	X		X	X	X					
CDI	X				X		X	X					X
SYN	X		X		X	X						X	
ACT	X		X	X	X	X				X			X
EMB	X		X	X						X		X	
CIE	X			X				X	X				X
DIFF	X	X										X	X
MEI	X	X	X				X		X		X		X
CART	X		X			X		X		X	X	X	

Figure V-15. Exploration des incertitudes par complexité dans DSM P_i

	PAM	PMIV	AF VOL	AF COM	CdP ACT	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB
PAM		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
PMIV	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X
AF VOL	X	X		X				X			X			X		
AF COM	X	X	X		X	X	X				X			X		
CdP ACT	X	X		X			X				X		X			
CdP SYN	X	X		X			X									
CdP CDI	X	X		X	X	X										
AF TPU	X	X	X						X	X						
CdP DIFF	X	X						X								
CdP MEI	X	X						X								
AF COU	X	X	X	X	X							X	X	X		
CdP EMB	X	X									X		X			
CdP CIE	X	X			X						X	X				
AF ELU	X	X	X	X							X				X	X
CdP CART	X	X												X		X
CdP LUB	X	X												X	X	

Figure V-16. Exploration des incertitudes par complexité dans DSM A_i

	PAM	PMIV	AF VOL	AF COM	CdP ACT	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB
FS VOL	X	X	X	X				X			X			X		
FS TPU	X	X						X	X	X				X		
FS ELU	X	X	X	X										X	X	X
FS COU	X	X			X						X	X	X			
FS COM	X	X	X	X	X	X	X									
CDI	X	X		X	X	X	X									
SYN	X	X		X		X	X									
ACT	X	X		X	X		X				X		X			
EMB	X	X									X	X	X			
CIE	X	X			X						X	X	X			
DIFF	X	X						X	X							
MEI	X	X						X		X						X
CART	X	X			X					X				X	X	X

Figure V-17. Exploration des incertitudes par complexité dans DSM P_i-A_i

5.3.3. Exploration des incertitudes par variabilité

Après avoir identifié tous les couplages dans cette situation intermédiaire, nous pouvons passer à la caractérisation des couplages dans les deux DSM et dans la MI. La levée des incertitudes par variabilité ne concerne pas uniquement les nouveaux couplages identifiés mais aussi les anciens (effets systémiques possibles).

	FS VOL	FS TPU	FS ELU	FS COU	FS COM	CDI	SYN	ACT	EMB	CIE	DIFF	MEI	CART
FS VOL		7	8	7	9	6	7	9	6	4	7	8	8
FS TPU	7		7								8	9	
FS ELU	8	7		7	6		5	5	6			9	9
FS COU	7		7		5			5	9	7			
FS COM	9		8	5		7	7	8					
CDI	6				7		9	8					6
SYN	7		5		7	9						8	
ACT	9		5	5	8	8				7			8
EMB	6		6	9						8		6	
CIE	4			7				7	8				5
DIFF	7	8										7	8
MEI	8	9	9						6		7		9
CART	8		9			6		8		5	8	9	

Figure V-18. Exploration des incertitudes par variabilité dans DSM P_i

	PAM	PMIV	AF VOL	AF COM	CdP ACT	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB
PAM		9	7	7	8	6	6	7	6	6	7	6	6	7	6	6
PMIV	9		7	7	9	8	8	7	8	8	7	8	8	7	8	8
AF VOL	7	7		7				7			6			8		
AF COM	7	7	7		9	9	9				3			7		
CdP ACT	8	9		9			9				5		5			
CdP SYN	6	8		9			9									
CdP CDI	6	8		9	9	9										
AF TPU	7	7	7						9	9						
CdP DIFF	6	8						9								
CdP MEI	6	8						9								
AF COU	7	7	6	3	5							9	9	7		
CdP EMB	6	8									9		8			
CdP CIE	6	8			5						9	8				
AF ELU	7	7	8	7				7			7				8	8
CdP CART	6	8												8		9
CdP LUB	6	8												8	9	

Figure V-19. Exploration des incertitudes par variabilité dans DSM A_i

	PAM	PMIV	AF VOL	AF COM	CdP ACT	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB
FS VOL	5	5	9	8				7			7			7		
FS TPU	5	5						9	8	8				5		
FS ELU	5	5	7	5										9	8	8
FS COU	5	5			5						9	7	7			
FS COM	5	5	7	9	9	8	8									
CDI	5	6		8	5	5	9									
SYN	5	6		8		9	5									
ACT	9	9		9	9		5				3		5			
EMB	5	6									7	9	5			
CIE	5	6			7						7	5	9			
DIFF	5	6						7	9	5						
MEI	5	6						7	5	9						8
CART	5	6			5					5				9	9	8

Figure V-20. Exploration des incertitudes par variabilité dans DSM P_i-A_i

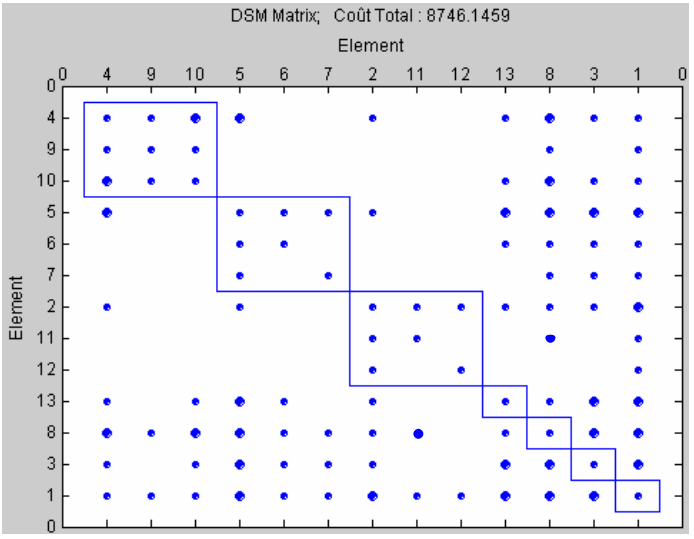
5.4. Simulation des architectures coévoluées

Les DSM et MI obtenues à la suite de l'exploration des incertitudes représentent des matrices totalement indépendantes les unes des autres, remplies par les acteurs du projet.

En utilisant le double processus flou présenté auparavant, nous avons obtenu deux DSM $DSMP_c$ et $DSMA_c$. On a par la suite appliqué le programme de clustering sur ces DSM en les filtrant au préalable.

5.4.1. Architecture coévoluée du produit

La figure V-21 montre l'architecture obtenue avec un seuil de filtrage de 3.2 et un IC= 0.8



1	2	3	4	5	6	7	8	9	10	11	12	13
FS VOL	FS TPU	FS ELU	FS COU	FS COM	CDI	SYN	ACT	EMB	CIE	DIFF	MEI	CART

Figure V-21. Architecture coévoluée du produit ($DSMP_c$)

L'architecture visualisée ci-dessus suggère les remarques suivantes :

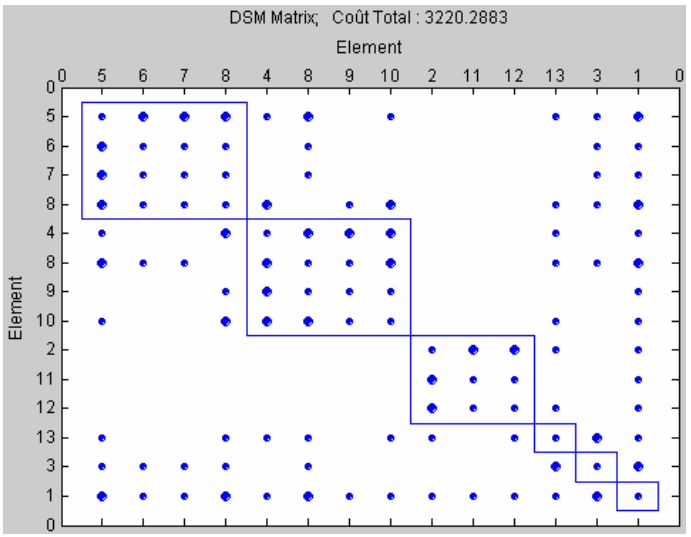
- L'algorithme a identifié trois modules et seulement deux éléments intégrateurs (FS VOL et ACT). Au lieu d'abaisser le seuil d'IC, nous avons opté pour la caractérisation manuelle de FS ELU et de CART comme intégrateurs.
- Concernant les éléments intégrateurs identifiés automatiquement, la FS VOL n'a pas changé de rôle comparativement à la situation initiale, ce qui sous-entend que les modifications introduites ne remettent pas en cause le rôle intégrateur de FS VOL. Ce résultat n'est pas étonnant puisque cette FS est toujours intégratrice.
- L'autre élément intégrateur identifié est l'actionneur. C'est le composant qui a été introduit pour réaliser une BV robotisée. A travers l'architecture obtenue, nous constatons que l'actionneur acquiert une importance très forte en devenant intégrateur sur toute la BV. Les acteurs du projet ne remettent pas en question ce rôle central, vu que le projet de reconception porte en majorité sur la conception de l'actionneur et des interfaces qui le lient aux autres composants. Cependant, il y a des composants et des fonctions qui n'interagissent pas avec l'actionneur.
- Concernant les modules identifiés, nous retrouvons les modules fonctionnels de la BV. Ce résultat est attendu puisque l'évolution technologique ne remet pas en cause le mode fonctionnement de la BV.

L'architecture globale de la BV représentée sur la figure V-21 a été construite en propageant les contraintes d'interaction entre les acteurs du projet représentées dans $DSMA_i$. Etant donné que les trois matrices construites après la levée des incertitudes sont indépendantes, la $DSMP_c$ que nous venons d'analyser est cohérente avec $DSMA_i$ et indépendante de $DSMP_i$.

5.4.2. Architecture finale du produit

Nous allons utiliser la méthode d'agrégation pour construire l'architecture finale du produit. Cette méthode calcule le barycentre entre les deux DSM du produit, celle construite directement par les acteurs et celle propagée à partir de l'organisation.

En utilisant a = ß = 0.5, nous obtenons la $DSMP_f$ dont l'architecture est représentée sur la figure V-22. Cette architecture est obtenue avec IC=0.75 et un seuil de filtrage à 2.6.



1	2	3	4	5	6	7	8	9	10	11	12	13
FS VOL	FS TPU	FS ELU	FS COU	FS COM	CDI	SYN	ACT	EMB	CIE	DIFF	MEI	CART

Figure V-22. Architecture finale du produit ($DSMP_f$)

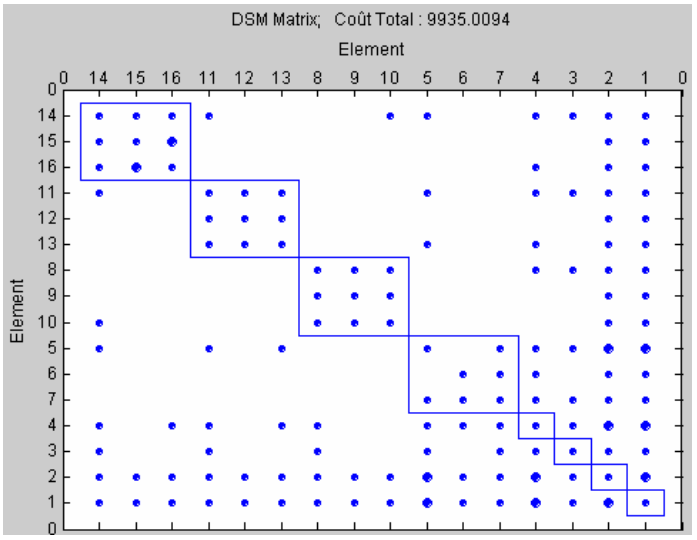
Nous analysons cette architecture de la manière suivante :

- L'architecture obtenue de la $DSMP_f$ est différente de celle obtenue dans $DSMP_c$. Ce qui signifie qu'il y avait bien une différence entre l'architecture perçue directement par les acteurs et celle obtenue par propagation de l'architecture de l'équipe de conception.
- La différence principale que nous obtenons avec $DSMP_f$ est que l'actionneur n'est plus intégrateur sur tout le produit mais seulement entre les modules fonctionnels couplage et commutation. Dans l'architecture représentée ci-dessus, l'actionneur est identifié comme faisant partie à la fois du module de commutation et de couplage. Ce rôle pivot reflète bien la mission affectée à l'actionneur, à savoir commander le changement de vitesse et l'embrayage.

L'architecture finale (figure V-22), telle que nous l'avons simulée, a reçu un meilleur accueil que celle obtenue directement par la coévolution (figure V-21). Cela appuie la nécessité de prendre en compte dans l'architecture finale du produit, l'architecture du produit simulée à partir de l'organisation et l'architecture intermédiaire du produit.

5.4.3. Architecture coévoluée des acteurs du projet

En partant de la $DSMP_i$ et de la MI P_i-A_i, nous obtenons la $DSMA_c$ représentée sur la figure V-23 après filtrage et clustering. Nous avons utilisé un seuil de filtrage de 2.3, un IC=0.8 et désigné AF VOL comme intégrateur manuellement.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PAM	PMIV	AF VOL	AF COM	CdP ACT	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB

Figure V-23. Architecture coévoluée des acteurs ($DSMA_c$)

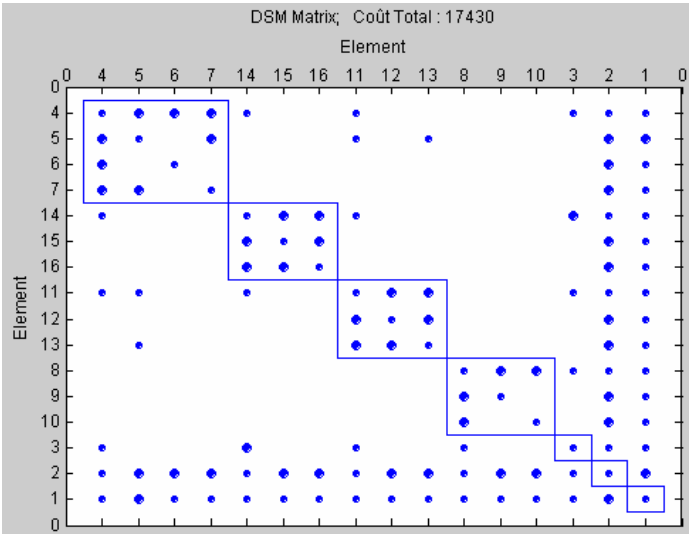
L'architecture ci-dessus suggère les remarques suivantes :

- L'algorithme de clustering a identifié quatre acteurs intégrateurs et quatre modules.
- Par comparaison à l'architecture initiale des acteurs de projet, nous retrouvons globalement les mêmes équipes de conception qui sont organisées par module fonctionnel du produit. Nous avons souligné dans l'analyse de l'architecture du produit que l'architecture fonctionnelle de la BV manuelle est conservée, d'où la conservation de la même organisation pour les acteurs du projet.
- La principale différence par rapport à l'architecture initiale est l'identification de l'architecte FS COM comme intégrateur sur toute l'équipe projet. L'architecture analysée ici est l'image de l'architecture intermédiaire du produit. Or dans cette architecture, la FS COM est à la fois couplée aux composants qui réalisent la commutation mais aussi aux autres fonctions systèmes de la BV robotisée.
- Nous remarquons aussi que le CdP ACT entretient un grand nombre d'interactions avec les autres acteurs du projet. Cependant, il appartient toujours au module commutation.

Nous allons utiliser la même méthode d'agrégation pour simuler l'architecture finale des acteurs du projet.

5.4.4. Architecture finale des acteurs de projet

En faisant la moyenne des $DSMA_c$ et $DSMA_i$, et en filtrant avec un seuil de 1.4, nous obtenons après clustering l'architecture représentée sur la figure V-24.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PAM	PMIV	AF VOL	AF COM	CdP ACT	CdP SYN	CdP CDI	AF TPU	CdP DIFF	CdP MEI	AF COU	CdP EMB	CdP CIE	AF ELU	CdP CART	CdP LUB

Figure V-24. Architecture finale des acteurs ($DSMA_f$)

En comparant l'architecture coévoluée et l'architecture initiale, nous remarquons qu'elles sont semblables. Ainsi, en proposant aux acteurs du projet d'arbitrer entre les différentes architectures obtenues, la réponse reçue est que même si les interfaces de certains composants sont modifiées, cela influe peu sur l'organisation des équipes mais cela engendre plus de négociations et plus de collaboration en amont entre les architectes fonctions systèmes (AF) et le responsable du projet (PAM).

Il est évident qu'une modification dans l'architecture du produit sans remise en question de son architecture fonctionnelle, n'est pas suffisante pour renverser la robustesse de l'organisation des acteurs de conception, et de ce fait de la remettre en question. Cela nous rappelle que les acteurs du projet partagent autre chose que le fait de travailler sur des composants qui ont des interfaces communes, ils sont liés par des connaissances et des compétences communes. Ce sont ces compétences qui donnent son identité à chaque module d'acteurs (acteur collectif).

6. Synthèse

Dans ce chapitre, nous avons proposé une méthode de coévolution des architectures des domaines du projet. Cette méthode de coévolution repose sur l'hypothèse qu'on parte d'une situation initiale cohérente pour aboutir à une situation finale qui l'est aussi.

Entre les situations initiale et finale, nous proposons de modéliser le besoin d'évolution par l'exploration des sources d'incertitudes. Cette exploration se base sur la typologie des incertitudes que nous avons adoptée et qui les classe en trois types : Incertitudes par Ambigüité, Incertitude par Complexité et Incertitude par Variabilité.

L'exploration des incertitudes rompt la cohérence entre les architectures de la situation initiale, nous proposons alors une méthode de coévolution qui permet de propager les contraintes de couplage d'un domaine vers un autre. Cette méthode de coévolution repose sur un traitement flou que nous avons présenté dans chapitre IV.

Pour appliquer la méthode de coévolution, nous avons étudié la situation de développement d'une BV robotisée à partir d'une BV manuelle.

Les résultats obtenus ont été soumis à l'appréciation de l'équipe de conception qui les a trouvés satisfaisants. Ce résultat appuie la pertinence de mettre à la disposition des chefs de projet et des architectes systèmes des outils qui permettent de simuler l'évolution des architectures en prenant en compte le couplage mutuel qui existe entre elles.

CONCLUSION GENERALE

CONCLUSION GENERALE

Dans cette partie, nous présentons les conclusions et perspectives de notre travail. Dans un premier temps, nous porterons notre attention sur notre contribution à la problématique traitée. Ensuite, nous tirerons un bilan de nos travaux du point de vue application Nous ouvrirons finalement les perspectives de cette recherche.

Au premier chapitre, nous avons montré que, dans le cadre de l'Ingénierie Système, un Produit a deux vues. L'extension de cette décomposition au projet de conception d'un produit, nous a permis d'affirmer que le projet a une vue fonctionnelle qui est celle des processus et une vue organique qui est celle des acteurs du projet.

Parallèlement à ce travail sur l'IS, nous avons développé notre positionnement vis-à-vis du concept d'architecture. Nous avons alors adopté comme typologie des architectures celle qui est proposée par Ulrich [1995] tout en affirmant qu'il y a un continuum entre une architecture totalement modulaire et une autre totalement intégratrice. Pour formaliser ce continuum, nous avons abandonné la caractérisation de l'architecture du produit par l'allocation des fonctions aux composants pour adopter une vision interne à chaque domaine, basée sur l'identification des interactions entre éléments d'un même domaine. Dans ce cadre, nous avons défini le concept de module comme un ensemble d'éléments fortement couplés les uns aux autres et le concept d'intégrateur comme étant un élément qui ne peut appartenir à aucun module puisqu'il est couplé à plusieurs d'entre eux. Le continuum se matérialise alors par l'existence d'architectures hybrides composées à la fois de modules et d'éléments intégrateurs.

En rendant la définition de l'architecture indépendante du système considéré, nous avons pu la généraliser au produit et à l'organisation du projet. On a admis alors que l'architecture du produit se compose deux architectures : l'architecture fonctionnelle et l'architecture organique. De même, l'architecture de l'organisation du projet est composée de l'architecture des processus et de l'architecture des acteurs.

En avançant dans notre démarche de formalisation, nous avons introduit dans le chapitre II les outils matriciels de représentation des architectures. Ces outils sont principalement les Matrices d'Incidence et les DSM qui permettent de caractériser les couplages inter et intra-domaines.

Dans le chapitre III, nous avons présenté l'algorithme qui va réaliser l'identification des architectures de tout domaine en partant de sa représentation matricielle sous la forme d'une DSM. Notre algorithme de clustering se distingue de ses prédécesseurs par sa capacité à identifier les éléments intégrateurs d'une façon semi-automatique, en relayant la vision de l'architecture système. Il a l'avantage aussi d'être plus performant dans l'identification de l'architecture « optimale » avec un fort degré de reproductibilité de la solution optimale. Enfin, en incorporant des indicateurs de modularité initialement développée par Whitfield et al. [2002], nous avons montré que notre algorithme est moins sensible à la valeur relative des interactions.

En nous positionnant dans les phases amont du processus de conception des produits complexes, nous avons voulu cibler les situations de conception où l'information est limitée. Ainsi, dans le chapitre IV, nous avons identifié quatre situations de conception de l'architecture des domaines du produit. Dans ces situations, nous avons voulu mettre en avant la capacité de la méthode que nous proposons à générer les DSM des domaines. Nous avons développé alors un traitement flou qui permet de créer deux DSM en partant d'une MI et de propager à travers une MI les contraintes de couplage d'un domaine donné vers un autre.

Afin de compléter les méthodes développées dans le chapitre IV et pour prendre en compte la nécessité de faire évoluer ensemble les architectures des domaines couplés, nous avons introduit dans le chapitre V le concept de coévolution des architectures. Nous avons formalisé cette coévolution comme étant la transition d'une situation initiale caractérisée par des architectures cohérentes vers une situation intermédiaire où l'on perd la cohérence des architectures pour arriver à une situation finale où les architectures retrouvent leur cohérence.

Dans ce travail, la situation intermédiaire est le résultat de l'exploration des incertitudes qui accompagnent l'introduction de changements dans le projet. Pour adapter l'exploration des incertitudes à nos outils, nous avons adopté une typologie des incertitudes qui s'applique aux outils matriciels : MI et DSM.

L'outil de développement des architectures que nous avons présenté dans ce mémoire est étroitement lié au cadre industriel. Les différentes situations de construction des architectures du produit sont ainsi inspirées des difficultés que nous avons rencontrées dans la récolte des informations nécessaires à notre méthode.

La méthode d'identification des architectures du produit a été appliquée à un projet de conception d'un moteur diesel. Dans ce cas, nous avons pu adapter notre méthode aux informations disponibles et nous avons ainsi pu fournir aux architectes systèmes des simulations des architectures fonctionnelles et organiques du moteur qu'ils ont validées.

Quant à la méthode de coévolution, elle a été appliquée à un projet de reconception d'une boîte de vitesse manuelle pour en faire une boîte robotisée. Nous avons, dans ce cas, simulé la coévolution de l'architecture du produit et des acteurs du projet après construction des nouvelles DSM par exploration des d'incertitudes.

D'autres MI et DSM ont été construites au cours du projet, mais non représentées dans ce mémoire (MI Produit-Tâches, MI tâches-Acteurs, DSM compétences).

Les travaux traitant de cette problématique sont très rares dans la littérature. Nous pensons que cela ouvre des voies de recherche intéressantes. Nous, présentons maintenant des limites et des pistes d'amélioration de nos travaux.

Ainsi, concernant la typologie d'architecture, nous avons remarqué que rares sont les travaux qui formalisent le rôle et la place des éléments intégrateurs dans une architecture. Jusqu'à aujourd'hui, ces éléments étaient considérés comme étant nuisibles et les algorithmes de clustering sont construits pour la plupart dans le but de les éliminer. A notre niveau, nous avons semi-automatisé l'identification de ces éléments, mais nous pensons qu'à terme il est possible de rendre automatique et dynamique la caractérisation et donc l'identification des éléments intégrateurs.

Dans ce travail, nous avons aussi considéré les couplages comme l'existence de tout type de contrainte entre un élément et un autre. Il est possible dans des travaux futurs de réutiliser les typologies d'interactions dans le produit [Pimmler et Eppinger, 1994] et celles de l'organisation [Sosa et al., 2004]. Ce travail nous permettra de construire plusieurs vues des contraintes entre les architectures des domaines du produit et de l'organisation du projet.

Une autre perspective liée au cadre conceptuel adoptée est la prise en compte des processus sous une forme statique ou temporelle pour simuler et modéliser le projet dans sa globalité.

Concernant l'algorithme de clustering, puisque d'une part une fonction objectif ne peut être qu'une approximation du raisonnement humain et que d'autre part, l'identification des éléments intégrateurs est encore subjective, nous pensons qu'il serait opportun de générer une famille d'architecture k-optimale (à k% de l'optimum ou en nombre k). Ainsi, nous considérons la situation où l'architecte peut tenir compte d'autres contraintes ou critères que ceux formalisés dans l'algorithme.

Pour rester dans le cadre de l'algorithme de clustering et de la méthode de construction des DSM, nous pensons avec le recul qu'il est possible d'énoncer des règles plus restrictives quant à la construction des MI et DSM. Ceci nous permettra alors d'éviter d'utiliser le filtrage dans les DSM. Ce problème peut être considéré d'un autre point de vue aussi, en réduisant par exemple la sensibilité de l'algorithme de clustering à la densité des DSM et en étudiant la robustesse des architectures simulées.

Enfin, nous pensons dans un futur très proche développer une plateforme logicielle (cas d'utilisation, scénarios, …) à partir des algorithmes et méthodes développés.

REFERENCES BIBLIOGRAPHIQUES

REFERENCES BIBLIOGRAPHIQUES

[AFIS] http://www.afis.fr/

[Akao, 1990] Akao, Y. *QFD-Integrating Customer Requirements into Product Design*, Productivity Press, 1990.

[Alexander, 1964] Alexander, C., *Notes on the Synthesis of Form*, Harvard Press, Boston, MA, 1964

[Altus et al., 1996] Altus, S., Kroo, I., Gage, P., "A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems," *ASME Journal of Mechanical Design*, 118,1996 ,

[Andersson et Sellgren, 2004] Andersson S. et U. Sellgren, "Representation and unse of functional surfaces", *7th Workshop on product structuring _Product Plateform Development*, Chalmers University of Technology, Goteborg, March 24-25, 2004.

[Avak, 2006] Avak B., "Life cycle adaptation of modular product families," *1st Nordic Conference on Product Lifecycle Management*, Goteborg, 2006.

[Balachandra, 2002] Balachandra, R., "Modular Design and Technological Innovation: The Case of the Hard Disk Drives", *Report 2002-02, The Information Storage Industry Center*, University of California, http://isic.ucsd.edu, December, 2006.

[Baldwin et Clark, 1997] Baldwin, C. et Clark, K., *Managing in the Age of Modularity.* Harvard Business Review . September-October: 84-93, 1997

[Baldwin et Clark, 2000] Baldwin, C. Y. et Clark, K. B. *Design Rules: The Power of Modularity Design*. MIT Press. pp.471. ISBN 0262024667, 2000.

[Baron, 2005] Baron C., *L'évaluation dans la conception système- Vers une aide à la conduite de projet*, HDR, Institut National Polytechnique de Toulouse, 2005

[Berge, 1958] Berge, C. *Théorie des Graphes et ses Applications*. Dunod, Paris, 1958.

[Bernard, 2000] Bernard A. « Modèles et approches pour la conception et la production intégrée » *APII – JESA* N° 34/ 2000.

[Bezdek et Pal, 1992] Bezdek, J.C., S.K. Pal, "Fuzzy Models for Pattern Recognition : Methods That Search for Structures in Data", *IEEE Press*. 1992.

[Blackenfelt et Sellgren, 2000] Blackenfelt M. et U. Sellgren "Design of robust interfaces in modular products", *DETC00, Proceedings of the 2000 ASME Design Engineering Technical Conferences*, September 10-13, 2000, Baltimore, Maryland

[Blackenfelt, 2000] Blackenfelt M. "Modularization by Relational Matrices - a Method for the Consideration of Strategic and Functional Aspects", *Proceedings of the 5th WDK Workshop on Product Structuring*. January 2000, Edt. Riithahuhta A., Pulkkinen A., Springer-Verlag

[Blackenfelt, 2001] Blackenfelt, M., *Managing complexity by product modularization*, Doctoral Thesis, Dept. of Machine Design, Royal Institute of Technology, Stockholm, 2001.

[Blanco, 1998] Blanco E., *L'émergence du produit dans la conception distribuée*, Thèse de Doctorat, Laboratoire 3S, INP Grenoble, 1998.

[Bohm et al, 2004] Bohm M. R. et R. B. Stone., "Representing Functionality To Support Reuse: Conceptual and Supporting Functions", *Proceedings of DETC '04 ASME 2004* Salt Lake City, Utah USA, September 28 – October 2, 2004

[Bonjour et Dulmet, 2006] Bonjour E., Dulmet M., "Pilotage des activités de conception par l'Ingénierie Système", *Ingénierie de la conception et cycle de vie du produit*, sous la direction de Roucoules L., éd.

Hermès, janvier 2006, pp. 85-105

[Bouchikhi, 1990] Bouchikhi H., Structuration des organisations : concepts constructivistes et étude de cas, *Economica*, Gestion, Paris, 1990.

[Braha, 2002] Braha D., "Partitioning Tasks To Product Development Teams", *Proceedings of DETC'02 ASME 2002 International Design Engineering Technical Conferences* Montreal, Canada, September 29-October 2, 2002

[Browning et Eppinger, 2002] Browning, T.R., Eppinger S.D. 'Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development', *IEEE Transactions on Engineering Management*, Vol. 49, No. 4, 443-458, 2002

[Browning, 1998] Browning TR. Use of dependency structure matrices for product development cycle time reduction, *Proceedings of the 5.fth ISPE international conference on concurrent engineering: research and applications*, Tokyo, Japan; 1998.

[Browning, 1999] Browning, T. R. "Designing System Development Projects for Organizational Integration." *Systems Engineering* 2(4): 217-225, 1999.

[Browning, 2001] Browning, T.R. 'Applying the design structure matrix to system decomposition and integration problems: A Review and New Directions', *IEEE Trans. Eng. Mgt*., Vol. 48, No .3, pp. 292-306, 2001.

[Carrascosa et al., 1998] Carrascosa, M., Eppinger, S. D. et Whitney, D. E. "Using the design structure matrix to estimate product development time', in *Proc. ASME Des. Eng. Tech. Conf. (Design Automation Conf.)*, Atlanta, GA, 1998.

[Cerny, 1985] Cerny V., "Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm," *J. Optimization Theory and Applications*, vol. 45, pp. 41-51, 1985.

[Chapman et Ward, 1997] Chapman C. B. et Ward S., *Project Risk Management: Processes, Techniques And Insights*. Chichester, UK, John Wiley, ISBN 0-471-95804-2, 1997.

[Chen et Lin, 2003] Chen, S.J. et Lin, L. 'Decomposition of interdependent task group for concurrent Engineering', *Computers et Industrial Engineering*, Vol. 44, pp. 435–459, 2003.

[Chen et Liu, 2005] Chen, K.M, Liu, R.J., 'Interface strategies in modular product innovation', *Technovation*, Vol. 25, pp. 771–782, 2005.

[Cho et Eppinger, 2001] Cho S-H, Eppinger S.D., "Product Development Process Modeling Using Advanced Simulation", *Proceedings of DETC'01 ASME 2001 Design Engineering Technical Conferences* Pittsburgh, Pennsylvania September 9-12, 2001

[Clarkson et al., 2004] Clarkson, P.J., Simons, C.S. et Eckert, C.M. 'Predicting change propagation in complex design', in *ASME Journal of Mechanical Design*, Vol. 126, No. 5, pp.765-797, 2004.

[Cohen et al., 1994] Cohen J. et Stewart I., The collapse of chaos. New York: Viking Penguin. 1994.

[Crawley et al., 2004] Crawley, E., De Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D., et Whitney, D. "The influence of architecture in engineering systems." Paper presented at *the MIT Engineering Systems Symposium*. Cambridge, MA. March 29-31, 2004. http://esd.mit.edu/symposium/monograph.

[D'Souza et Simpson, 2003] D'Souza, B., et Simpson, T., "A Genetic Algorithm Based Method for Product Family Design Optimization," Engineering Optimization, Vol. 35(1) , 2003.

[Dahmus et al., 2000] Dahmus J. B., Gonzales-Zugasti J. P., et Otto K. N., "Modular Product Architecture", *Proc. of DETC 00: ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Baltimore, MD, 2000.

[David et al., 2002] David M., Z. Idelmerfaa et J. Richard, "Organization method for complex cooperative design projects." *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, Hammamet, 2002.

[Demiriz et al., 1999] Demiriz, A, Bennett, K. P., Embrechts, M. J., "Semi-Supervised Clustering using Genetic Algorithms". *Proceedings of Artificial Neural Networks in Engineering*, 1999.

[Dronkov et al., 1993] Dronkov D., Reinfrank M., Helledom H., *An introduction to fuzzy Control*, Springer-Verlag,

London 1993.

[Duray, 2000] Duray, R., "Approaches to mass customization: configurations and empirical validation", *Journal of Operations Management*, 18 (2000) 605-625

[EIA 632] EIA 632 : Processes for Engineering a System, Avril 1998.

[Eppinger et al., 1994] Eppinger, S. D., Whitney, D. E., Smith, R. P. et Gebala, D. A. 'A Model-Based Method for Organizing Tasks in Product Development', *Research in Engineering Design*, Vol. 6, No.1, pp.1-13, 1994.

[Eppinger et Salminen, 2001] Eppinger, S.D., et Salminen, V., 'Patterns of product development interactions', *Int. Conf. on Engineering Design, ICED 01*, Vol. 1, Glasgow, August 21-23, pp.283-290, 2001.

[Ericsson et Erixon, 1999] Ericsson, A. et Erixon, G. *Controlling design variants: Modular product platforms*. ASME press, New York, NY. pp145. 1999. ISBN 0-87263-514-7.

[Erixon, 1996] Erixon, G., "Modular Function Deployment (MFD), Support for Good Product Structure Creation", *2nd WDK Workshop on Product Structuring*, June 3-4 -96 in Delft, Holland, 1996.

[Erixon, 1998] Erixon G. *Modular Function Deployment - A Method for Product Modularisation*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 1998.

[Eynard, 1999] Eynard B., *Modélisation du produit et des activités de conception, contribution à la conduite et à la traçabilité des processus de conception*, Thèse de Doctorat, Université de Bordeaux, 1999.

[Fernandez, 1998] Fernandez, C., "Integration Analysis of Product Architecture to Support Effective Team Co-location" MS Thesis. Massachusetts Institute of Technology, 1998.

[Fitzpatrick et Askin, 2005] Fitzpatrick, E.L. et Askin, R.G., 'Forming effective worker teams with multi-functional skill requirements', *Computers et Industrial Engineering*, Vol. 48, pp.593–608, 2005.

[Forest et al., 2005] Forest J., Méhier C., Micaëlli J.P., *Pour une science de la conception*, collection sciences humaines et technologie, UTBM, Belfort 2005.

[Galbraith, 1994] Galbraith, J. R., *Designing Organizations: an Executive Briefing on Strategy, Structure, and Process*, Jossey -Bass Inc., San Fransisco, 1994.

[Galbraith, 1977] Galbraith, J.R., *Organization Design* Addison-Wesley, Reading, Mass, 1977.

[Génelot, 1992] Génelot D., *Manager dans la complexité, réflexions à l'usage des dirigeants*, édition INSEP, Paris 1992.

[Gershenson et al., 2004] Gershenson, J. K., Prasad, G. J., et Zhang, Y. "Product modularity: measures and design methods". *Journal of Engineering Design*. Vol 15. No1. pp. 33-51. Feb 2004.

[Giard, 1991] Giard V., *Gestion de projets*, Economica, 160p., 1991.

[Girard, 2004] Girard P. et Doumeingt G., «Modelling of the engineering design system to improve performance », *I.J. Computers et Industrial Engineering*, Vol. 46/1 p.43-67, 2004.

[Goldberg, 1989] Goldberg, D. E., *Genetic Algorithms in Search, optimization, and Machine Learning*, Addison-Wesley, New York, NY, 1989.

[Graham, 1991] Graham I.. "Fuzzy logic in commercial expert systems–resultsand prospects." *Fuzzy Sets and Systems*, 40(3):451–72, April 1991.

[Guivarch, 2003] Guivarch A.D. *Concurrent Process Mapping, Organizations, Project and Knowledge Management in Large-Scale Product Development Projects Using the Design Structure Matrix Method.* MS at the Massachusetts Institute of technology, 2003.

[Gulati et Eppinger, 1996] Gulati R. K. et S. D. Eppinger, *"The Coupling of Product Architecture and Organizational Structure Decisions."* Massachusetts Institute of Technology, Working Paper Number 3906, 1996.

[Guo et Gershenson, 2004] Guo, F. et Gershenson, J. K. "A comparison of modular product design methods on improvement and iteration". *In Proc of ASME Design Engineering Technical Conferences*. Salt Lake City, UT. 2004.

[Gutiérez-Estrada, 2007] Gutiérrez-Estrada C. *Méthodes et outils de la conception système couplée à la conduite de projet*, Thèse, INSA Toulouse, le 06 Février 2007

[Harani, 1997] Harani Y., *Une approche multi modèles pour la capitalisation des connaissances dans le domaine de la conception*, Thèse de l'Université de Grenoble, 1997.

[Harel, 1987] Harel, D., "Statecharts : a visual formalism for complex systems." *Science of Computer Programming*, 8, p. 231–274, 1987.

[Harmel et al., 2006a] Harmel G., E. Bonjour et M. Dulmet «*Architecture des produits et des organisations : modélisation et pilotage par l'incertitude.* Actes de la 6ème Conférence Francophone MOdélisation et SIMulation, MOSIM'06, CD ROM, 9 pages, 3-5 avril 2006, Rabat, Maroc.

[Harmel et al., 2006b] Harmel G., E. Bonjour et M. Dulmet, *"Architecture des systèmes complexes: modélisation et pilotage par l'incertitude"*, 4è congrès annuel de l'AFIS, actes sur CD, Toulouse, mai 20

[Harmel et al., 2006c] Harmel G., Bonjour E. et Dulmet M., *"Products and organisation architecture modeling and control: from strategic expectations to strategic competencies"*, 12th IFAC Symposium on INformation COntrol problems in Manufacturing, INCOM, IFAC/IEEE/IFIP, CD ROM, 6 pages, Saint-Etienne, May 2006.

[Harmel et al., 2006d] Harmel G., E. Bonjour et Dulmet. *"A method to manage the co-evolution of product and organization architectures"*, IMACS-IEEE Multiconference on Computational Engineering in Systems Applications, CESA 2006, October 4-6, Beijing, China

[Harmel et al., 2007] Harmel G., E. Bonjour et M. Dulmet. *«A fuzzy method to design both functional and physical product architectures"*, International Conference on Engineering Design, ICED'07, 28 - 31 AUGUST 2007, PARIS, FRANCE, communication acceptée.

[Hartigan, 1975] Hartigan, J.A. *Clustering Algorithms*, John Wiley & Sons, N.Y, 1975.

[Hatchuel et Weil, 2002] Hatchuel A. et Weil B. « La théorie C-K, Fondement et usage d'une théorie unifiée de la conception », *International Conference Sciences of Design*, Lyon, 2002.

[Hatley et Pirbhai, 1987] Hatley et Pirbhai. *Strategies for real-time system specification*. Dorset House, 1987

[Herroelen et Leus, 2005] Herroelen, W. et Leus, R. 'Project scheduling under uncertainty —survey and research potentials', Research report 0225, Department of Applied Economics, K.U. Leuven, *European Journal of Operational Research*, Vol. 165, pp.289–306, 2005

[Holtta et al., 2005] Hölttä K., Suh E., et De Weck O. "Tradeoff Between Modularity and Performance for Engineering Systems and Products" *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN ICED 05 MELBOURNE*, AUGUST 15 – 18, 2005

[Holtta et Salonen, 2003] Hölttä K., Salonen M. Comparing three different modularity methods. *Proc. of DETC'03, Design Engineering Technical Conferences, ASME 2003*, Chicago, USA, September 2003.

[Hölttä, 2005] Hölttä, K. (2005) *Modular product platform design*. PhD, Helsinki University of Technology, Espoo, Finland, ISBN 951-22-7766-2.

[Huang et Kusiak, 1998] Huang C.C., A. Kusiak. "Modularity in design of products and systems", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 28, No. 1, 1998, pp. 66-77.

[Huang, 2000] Huang, C.C.: "Overview of modular product development", *Proc. Natl. Sci. Council, ROC(A)*, vol. 24, N.3, 2000, pp.149-165.

[Hubka et Eder 1988] Hubka V. et E. Eder, *Theory of Technical Systems*, Springer-Verlag, New York, 1988.

[Idicula, 1995] Idicula, J., *Planning for Concurrent Engineering*, Thesis draft, Nanyang Technology University, Mars, 1995.

[IEEE 1220] IEEE 1220: Standard for application and Management of the Systems Engineering Process, 1999.

[ISO 15288] ISO 15288: Systems Engineering – System Life-Cycle Processes, AFNOR Z 67-288 (Ingénierie systèmes – Processus de cycle de vie des systèmes), Novembre 2003.

[Jiao et al., 2000] Jiao J., Tseng M., M., Zou Y., «Generic bill of materials and Operations for high-variety

production management », *Concurrent Engineering: Research and Applications - CERA*, vol. 8, n°4, 2000, p. 297-322

[Jiao et al., 2006] Jiao, J., Simpson, T., et Siddique, Z. "Product family design and platform-based product development: a state-of-the-art review", *Journal of intelligent manufacturing, special issue on Product family design and platform-based product development*, 2006.

[Kandel, 1992] Kandel A., editor. *Fuzzy expert systems*. CRC Press, Boca Raton, Florida, 1992.

[Keller et al., 2005] Keller, R., Eckert C.M. et Clarkson P.J., 'Multiple views to support engineering change management for complex products' in *3rd International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV '05), London, UK*, 2005.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C.D. et Vecchi, M.P., Optimisation by simulated annealing, *Science*, **220**(4598):671-680, 1983

[Krishnan et Gupta, 2001 Krishnan, V. et Gupta, S., "Appropriateness and impact of Platform-Based Product Development", *Management Science*, 47, 1, pp. 52-68, 2001

[Kusiak et al., 1993] Kusiak, A. et E. Szczerbicki, "Transformation from Conceptual to Embodiment Design, *IEEE Transactions*, Vol 25, No.4, 1993

[Kusiak et Huang, 1996] Kusiak, A. et Huang, C., "Development of modular products," *IEEE Transactions on Components, Packaging and Manufacturing technology*- Part A, 19(4) 523-538, 1996

[Kusiak et Wang, 1993] Kusiak A., Wang J., *Decomposition in Concurrent Design, Engineering, in: Concurrent Engineering*, J. Wiley, 1993.

[Langlois et Robertson, 1992] Langlois, R. N. et P. L. Robertson "Networks and innovation in a modular system: lessons from the microcomputer and stereo component industries." *Research Policy*, 21(4), 297-313, 1992.

[Larses et Blackenfelt, 2003] Larses O. et Blackenfelt M., "Relational reasoning supported by quantitative methods for product Modularization" *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN ICED 03* STOCKHOLM, AUGUST 19-21, 2003

[Larses, 2005] Larses O., "Applying quantitative methods for architecture design of embedded automotive systems." *Proc.INCOSE International Symposium 2005*. Rochester, NY. July 10-15. 2005.

[Lartigue, 2003] Lartigue N., « Compétence et conception », *congrès SIA*, Poissy, France, 09-2003.

[Lawson et Karandikar, 1994] Lawson M. et Karandikar H., « A Survey of CE », *CERA Journal* 2 : 1-6., 1994

[Lefebvre et al., 2002] LEFEBVRE P., ROOS P., SARDAS J-C., «From the management of expertise to the management of design metier », *EURAM Conference*, Stockholm, Suède, 2002.

[Levine, 2005] Levine, H. A., *Project Portfolio Management: A Practical Guide to Selecting Projects, Managing Portfolios and Maximizing Benefits*, Jossey-Bass, 2005

[Loch et al., 2000] Loch C.H., M.T. Pich et A. De Meyer, «Project uncertainty and management styles » *INSEAD R&D 2000/31/TM/CIMSO 10*. Fontainebleau, France, 2000

[Lockledge et Salustri, 2001 Lockledge J. C. et F. A. Salustri "Design Communication Using a Variation of the Design Structure Matrix" *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN ICED 01* GLASGOW, AUGUST 21-23, 2001

[Maier et Rechtin, 2000] Maier, M. W. et Rechtin, E. *The art of systems architecting*. 2nd ed. CRC Press 2000. ISBN 0-8493-0440-7.

[Malmqvist, 2002] Malmqvist, J., 'A classification of matrix based methods for product modelling' *Design 2002*, 14-17, Cavtat-Dubrovnik, Croatia, 2002

[McCord et Eppinger, 1993] McCord, K. R. et Eppinger, S. D., '*Managing the Integration Problem in Concurrent Engineering*', M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no.3594, 1993

[McCulley et al., 1996] McCulley, C., et Bloebaum, C., "A Genetic Tool for Optimal Design Sequencing in Complex

Engineering Systems," *Structural Optimization*, 12, pp. 186-201, 1996

[Meinadier, 2002] Meinadier, J-P., *Le métier d'intégration de systèmes*, Hermès Sciences Publications, 2002

[Menand, 2002] Menand S., Modélisation pour la réutilisation du processus de conception multi acteur de produits industriels, Thèse de Doctorat, Grenoble, 2002.

[Meridith et al., 1995] Meridith J.R. et Mantel S.J., *Project management-a managerial approach*, third edition, New York : Wiley. 1995.

[Messac et al., 2002] Messac, A., Martinez, M. et Simpson, T., "Introduction of Product Family Penalty Function Using Physical Programming," *ASME Journal of Mechanical Design*, Vol. 124, 164-172, , 2002

[Micaëlli et Forest, 2003], Micaëlli J.P. et Forest J., *Artificialisme, Introduction à une théorie de la conception*, PPUR, Lausanne, 2003.

[Midler, 1998] MIDLER C, *L'auto qui n'existait pas, managements des projets et transformation de l'entreprise*, Dunod, Paris, 1998.

[Morelli et al., 1995] Morelli M.D., Eppinger S.D et R.K. Gulati, "Predicting Technical Communication in Product Development Organizations", *IEEE Transactions on Engineering Management*, Vol. 42 No. 3 August, 1995

[Morris et al., 1987], Morris P.W.G. et Hugh G.H., *The anatomy of major Projects*. Chichester : Wiley. 1987.

[Mtopi, 2006] Mtopi F.B. « Contribution à une méthodologie de conception modulaire : Modélisation de la diversité dans les familles de produits », thèse de l'Université de Franche-Comté, le 12 juillet 2006.

[Newcomb et al., 1998] Newcomb, P. J., Bras, B., et Rosen, D. W., "Implications of modularity on product design for the life cycle", *Journal of Mechanical Design*, Volume 120, 1998, pp. 483-490.

[Nightingale, 2000] Nightingale P., «The Product-process-Organization relationship in complex development projuects," *Res. Policy*, Vol 29, pp. 913-930, 2000

[OMG, 2006] OMG, "OMG Systems Modeling Language (OMG SysML) Specification", Final Adopted Specification, ptc/2006-05-04

[Oosterman, 2001] Oosterman, B. *Improving Product Development Projects by Matching Product Architecture and Organization*, PhD thesis, Gröningen University, The Netherlands, ISBN: 90-72591-99-2, 2001

[Otto et Wood, 2001] Otto, K. et Wood, K., 2001, *Product design: techniques in reverse engineering and new product development*, Prentice Hall, Upper Saddle River, NJ.

[Pahl et Beitz 1984] Pahl G. et W. Beitz, *Engineering Design*, Ken Wallace (Editor), London, 1984.

[Pahl et Beitz, 1996] Pahl G., Beitz W. *Engineering Design: a Systematic Approach*, 1996 (2nd Ed., Springer-Verlag, London).

[Penalva, 1997] Penalva J.M. "la modélisation par les systèmes complexes", thèse de doctorat, Université de Paris Sud, décembre 1997.

[Perrin, 1999] Perrin J., *Pilotage et évaluation des processus de conception*, L'harmattan, 1999.

[Perrin, 2001] Perrin J. (sous la direction de), *Conception entre science et art : regards multiples sur la conception*, Presses Polytechniques et Universitaires Romandes, Lausanne, 2001

[Pich et al., 2002] Pich, M., Loch, C. De Meyer, A., 'On uncertainty, ambiguity, and complexity in project management', *Management Sci.* Vol. 48, No. 8, pp.1008–1023, 2002

[Pimmler et Eppinger, 1994] Pimmler, T. U. et Eppinger, S. D., 'Integration Analysis of Product Decompositions', *Proceedings ASME Design Theory and Method Conference (DTM'94)*, Vol 68, pp. 343-351, 1994

[Pimmler, 94] Pimmler T., "A Development Methodology for Product Decomposition an Integration" MS Thesis. Massachusetts Institute of Technology, 1994

[PMI, 2001] Project Management Institute, "Practice standard for work breakdown structures", ISBN

1-880410-81-8, Library of Congress Cataloging, Pennsylvania, USA, 2001 (www.pmi.org).

[PMI, 2004] Project Management Institute, "A Guide to the Project Management Body of Knowledge (PMBOK® Guide)" - Third Edition, ISBN: 193069945X, 388p., 2004.

[Prasad, 1997] PRASAD B., *Concurrent Engineering Fundamentals*, vol 1et2, Prentice Halls, 1997.

[Rissien, 1978] Rissinen, J., "Modeling by Shortest Data Description," *Automatica*, 14, pp. 465-471, 1978

[Robin et al., 2004] Robin V., Girard P., Rose B., « Intégration des connaissances dans les environnements de conception pour le pilotage de la conception collaborative », *colloque C2EI*, actes CD, 10p, Nancy, AIP Lorrain, CRAN - ERPI, 1-2 décembre 2004.

[Rodriguez-Torro, 2004] Rodriguez-Torro C., Jared G. et Swift K., "Product Development complexity metrics: A framework dor proactive-*DFA implementation. International Design Conference- Design 2004*, Dubrovnik, 18-21 Mai, 2004

[Rogers et McCulley, 1996] Rogers J. et M. McCulley, «Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes », *NASA Technical Memorandum* TM-110247, 1996

[Roozenburg et Eekels, 1995] Roozenburg N. et J. Eekels, *Product design: fundamentals and methods*, Wiley & Sons, Chichester, 1995

[Sanchez et Mahomey, 1996] Sanchez, R. et Mahomey, J.T., 'Modularity, flexibility and knowledge management in product and organization design', *Strategic Management Journal*. Vol. 17, pp. 63-76, 1996

[Sanchez, 1999] Sanchez, R., "Modular architectures in the marketing process", *Journal of Marketing*, Vol 63 (1999) 92-112, 1999

[Sanchez, 2002] Sanchez, R., "Using modularity to manage the interactions of technical and Industrial design" *Design Management Journal*. 2 (2002) 8

[Sanderson et Uzumeri, 1990] Sanderson, S. W. et V. Uzumeri, *Strategies for New Product Development and Renewal: Design-based Incrementalsim*. Working Paper, Center for Science and Technology Policy, Rensselaer Polytechnic Institute, Troy, New York, NY, U.S.A, 1990

[Sharman et Yassine, 2004] Sharman, D.M. et Yassine, A., 'Characterizing Complex Product Architectures', *Systems Engineering* Vol. 7, No. 1, pp. 35-60, 2004

[Siddique et Rosen, 1999] Siddique, Z. et Rosen, D., "Product platform design: A graph grammar approach," *Proceedings ASME Design Engineering Technical Conferences*, Las Vegas, Nevada, DETC99/DTM-8762, 1999

[Simon, 1997] Simon H.A., *Sciences des systèmes, Sciences de l'artificiel*, version traduite en français par J-L. Le Moigne, Dunod, 1997.

[Smith et al., 1997] Smith, R.P. et Eppinger, S.D., "A Predictive Model of Sequential Iteration in Engineering Design" *Management Science*, vol. 43, pp. 1104-1120, 1997.

[Sosa et al., 2000] Sosa, M.E., Eppinger, S.D. et Rowles C.M., 'Designing modular and integrative systems', *Proceedings of DETC '00: ASME 2000 International Design Engineering Technical Conferences*, 2000

[Sosa et al., 2003] Sosa, M., Eppinger, S.D. et Rowles C. 'Identifying modular and integrative systems and their impact on design team interactions', *Transactions of the ASME*, Vol. 125, pp. 240-252, 2003

[Sosa et al., 2004] Sosa, M.E., Eppinger, S.D. et Rowles C.M., "The Misalignment of Product Architecture and Organizational Structure in Complex Product Development," *MANAGEMENT SCIENCE* Vol. 50, No. 12, December 2004, pp. 1674–1689, 2004

[Sosa, 2005] Sosa, M.E "A Network Approach to Define Modularity of Product Components", *Proceedings of ASME Design Theory and Methodology Conference*, Long Beach, CA, September 2005.

[Steward, 1981] Steward D.V., "The Design Structure System: A Method for Managing the Design of Complex Systems" *IEEE Transactions on Engineering Management*, vol. 28, pp. 71-74, 1981

[Stone et al., 1998] Stone, R, Wood, K, Crawford, R ÔA Heuristic Method to Identify Modules from a Functional Description of a Product *Proceedings of DETC98* DETC98/DTM-5642 Atlanta, GA ,1998

[Stone et al., 2000] Stone, R. B., Wood, K. L. et Crawford, R. H., "A heuristic method for identifying modules for product architectures", *Design Studies*, 21, 1, pp. 5-31, 2000

[Sudjianto et Otto, 2001] Sudjianto, A. et Otto, K. Modularization to support multiple brand platforms. *In Proc of ASME Design Engineering Technical Conferences*. Pittsburgh, PA.September 9-12, 2001.

[Suh, 1990] Suh, N. P., *Principles of Design*, Oxford University Press, Cambridge, UK, 1990

[Thebeau, 2001] Thebeau, R. E. *Knowledge Management of System Interfaces and Interactions for Product Development Processes*. Masters Thesis. System Design & Management Program, Massachusetts Institute of Technology. 2001.

[Trassaert, 2002] Trassaert T P., « Concevoir le produit, l'organisation et la stratégie – le métier de systémier », *International Conference in the Sciences of Design*, Lyon, in honour of Herbert Simon, 2002.

[Tseng et al., 2004] Tseng, T.L., Huang, C.C., Chu, H.W. et Gung, R.R., 'Novel approach to multi-functional project team formation', *International Journal of Project Management*, Vol. 22, pp.147–159, 2004

[Ulrich et Eppinger, 2000] Ulrich, K. T. et Eppinger, S. D. *Product Design and Development*. McGraw-Hill.2nd edition. 2000.

[Ulrich et Tung, 1991] Ulrich K. & Tung K. Fundamentals of Product Modularity. *In proc of ASME Winter Annual Meeting Symposium on Design and Manufacturing Integration*. pp.73-79. Atlanta, GA. November 1991.

[Ulrich, 1995] Ulrich, K.T., 'The Role of Product Architecture in the Manufacturing Firm', *Research Policy*, Vol. 24, pp.419-440, 1995

[Van Wie et al., 2001] Van Wie, M., Greer, J., Campbell, M., Stone, R. et Wood, K.,'Interfaces and Product Architecture', *ASME Design Engineering Technical Conference Proc., DETC01/DTM-21689*, 2001

[VDI 2206] VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme. Düsseldorf: VDI Verlag, 2004-06

[VDI 2221] VDI-Richtlinie 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. Düsseldorf: VDI Verlag, 1993-05

[Ward et al.,1995] Ward, W., J. F. Liker, J. J. Cristiano, et D. K. Sobek, " The second Toyota paradox: delaying decisions can make better cars faster". *Sloan Management Review*, 36(3), 43-61, 1995

[Ward et Chapman, 2003] Ward S., C.B. Chapman., "Transforming project risk management into project uncertainty management" *International Journal of Project Management* Vol 21, 97-105, 2003.

[Ward et Mellor, 1985] P. T. Ward et S. J. Mellor. *Structured development for real-time systems*, vol. 1-3 of Yourdon computing series. Prentice-Hall, 1985.

[Warfield, 1973] "Binary Matrices in System Modeling" *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 441-449, 1973.

[Whitfield et al., 2002] Whitfield, R.I., Smith, J.S. et Duffy, A.H.B., 'Identifying Component Modules', *In Proceedings of Seventh International Conference on Artificial Intelligence in Design (AID'02)*, Cambridge, United Kingdom, pp. 571-592, 2002

[Whitney, 2003] Whitney D. E. *Physical limits to modularity*. Massachusetts Institute of Technology, Engineering Systems Division. Working paper. ESD-WP-2003-01.03-ESD.

[Yang et al., 2004] Yang T.G., Beiter K.A. et Ishii K., "Product Platform Development: An Approach for Products in the Conceptual Stages of Design", *Proceedings of IMECE2004 ASME International Mechanical Engineering Congress* November 13-19, 2004, Anaheim, California USA

[Yannou et Bonjour, 2006] Yannou B. et E. Bonjour (sous la Dir. De), *Evaluation et décision dans les processus de conception*, Edition Hermès Sciences, Série Productique, collection Traité IC2, 2006, 251p

[Yassine et al., 2000] Yassine A.A., D. E. Whitney, J. Lavine, et T. Zambito, "Do-it-rightfirst-time (DRFT)

approach to DSM restructuring," in *Proc. ASME 2000 Int. Design Engineering Technical Conf. (DTM)*, Baltimore, MD, 2000.

[Yassine et. al., 2001] Yassine A., D. Whitney et T. Zambito " Assessment of rework probabilities for simulating product development processes using the design structure matrix (DSM)" *Proceedings of DETC '01 ASME 2001 International Design Engineering Technical Conferences Computers and Information in Engineering Conference* Pittsburgh, Pennsylvania, September 9-12, 2001

[Yassine, 2004] Yassine A.**,** "An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method", *Quaderni di Management (Italian Management Review)*, www.quaderni-di-management.it, No.9, 2004.

[Yu et al., 2003] Yu T., Yassine A. et Goldberg D., 'Genetic algorithm for developing modular product architectures', In *Proc ASME 15th Int Conf Des Theory Methodology*, Chicago, September, 2003

[Zadeh, 1975] Zadeh L.A., "The Concept of a Linguistic Variable and its Application to Approximate Reasoning" - *III. Information Science*, 9, pp. 43-80, 1975

[Zakarian et Rushton, 2001] Zakarian, A. et Rushton, G., "Development of modular electrical systems," *IEEE Transactions on Mechatronics* 6(4) 507-520, 2001

[Zamirowski et Otto, 1999] Zamirowski, E. J. et Otto K. N. "Identifying Product Family Architecture Modularity Using Function and Variety Heuristics". *In Proc of ASME Design Engineering Technical Conferences*. Las Vegas, NV. 1999.

ANNEXES

ANNEXES

Dans ces annexes, nous présentons une partie des algorithmes qu'on a développés. La programmation de l'algorithme de clustering (chapitre III), des méthodes d'identification des architectures (chapitre IV) et de la méthode de coévolution (chapitre V) a été réalisée sous Matlab.

L'algorithme mettant en œuvre la méthode de clustering est une évolution d'un algorithme existant dont la dernière version est celle de Thebeau [2001]. Nous nous sommes basés sur cet algorithme pour le modifier et développer les procédures propres à notre algorithme de clustering.

L'algorithme de Thebeau est composé des fonctions suivantes :

run_cluster	C'est la fonction principale de l'algorithme, elle fait appel aux autres fonctions pour réaliser le clustering et afficher les résultats
DSM	C'est le fichier où on saisit la DSM dont on veut identifier l'architecture
bid	C'est une fonction qui calcule les enchères des clusters vers un élément donné
cluster	C'est la fonction qui réalise le clustering en faisant appel à une fonction coût
coord_cost	C'est la fonction qui calcule le coût d'une architecture proposée par cluster
delete_clusters	Cette fonction supprime les clusters en double
reorder_cluster	C'est la fonction qui trie les clusters selon l'ordre décroissant des tailles
reorder_dsm_bycluster	C'est la fonction qui réordonne les clusters selon la matrice des clusters
dsm_autolabel	C'est la fonction qui assigne un nombre aux éléments de la DSM
place_diag	C'est la fonction qui place 1 sur la diagonale de la DSM
graph_matrix	C'est la fonction qui dessine les DSM utilisées (initiale, finale, …)
line_mult_cluster	C'est la fonction qui fait afficher une ligne pour les éléments dupliqués dans la DSM finale
plot_cluster_list	C'est la fonction qui imprime la liste des modules et des éléments qui les composent
likness_calc	C'est la fonction qui permet de lancer plusieurs fois l'algorithme de clustering pour comparer les architectures obtenues

Dans ce travail, nous avons apporté des modifications aux fonctions DSM, run_cluster, bid, coord_cost et cluster.

Dans cette annexe, nous présentons uniquement les 4 premières fonctions. La fonction DSM dans notre travail se compose de deux fonctions. Pour chacune des situations de construction

des architectures identifiées dans le chapitre IV, ainsi que pour la méthode de coévolution, nous définissons une fonction pour présenter les données et une autre pour mettre en œuvre le traitement flou quand il existe et les différentes opérations (filtrage, normalisation, etc…) qu'on réalise sur les DSM pour les préparer au clustering. Ces deux fonctions mettent en œuvre la construction des DSM dont on veut identifier l'architecture.

Dans cette annexe, nous détaillerons uniquement la méthode de construction dans la 2ème situation du chapitre IV et la méthode de coévolution.

Dans la suite de ces annexes, les commentaires sur les algorithmes sont précédés de %.

Construction de deux DSM à partir d’une MI

Deux fonctions (fichiers) :

- situation2_donnees.m : dans ce fichier, non saisissons les données relatives à la MI FS-COMP
- situation2_DSM.m : dans ce fichier, nous mettons en œuvre le traitement flou et les autres opérations pour aboutir aux DSM finales

```
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
%
%       Fichier:    situation2_donnees.m
%
%
%       Créé par:           G. HARMEL
%                           LAB
%                           Besançon FRANCE
%
%       Date: Juin 2006
%
%
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************


N_COMP = 15; % nous déclarons le nombre de composant dans la MI
N_FS = 13;      % nous déclarons le nombre de FS dans la MI

INCED = zeros(N_COMP,N_FS); % nous créons la MI

% nous saisissons la MI
INCED(1,1)=8;
INCED(1,3)=9;
INCED(1,9)=6;
INCED(1,10)=8;
INCED(1,11)=7;
INCED(1,13)=6;

INCED(2,2)=8;
INCED(2,4)=8;
INCED(2,6)=0;
INCED(2,7)=7;
INCED(2,8)=0;
INCED(2,9)=7;
INCED(2,10)=8;
INCED(2,11)=5;
INCED(2,13)=8;

INCED(3,1)=5;
INCED(3,2)=5;
INCED(3,3)=6;
INCED(3,4)=5;
INCED(3,6)=8;
INCED(3,7)=7;
INCED(3,9)=7;
INCED(3,11)=8;
INCED(3,13)=8;

INCED(4,1)=9;
INCED(4,2)=9;
INCED(4,3)=6;
INCED(4,4)=9;
INCED(4,7)=4;
INCED(4,8)=5;
INCED(4,9)=6;
INCED(4,10)=9;
INCED(4,11)=3;
INCED(4,13)=7;

INCED(5,1)=8;
```

```
INCED(5,3)=9;
INCED(5,6)=6;
INCED(5,7)=5;
INCED(5,9)=6;
INCED(5,10)=6;
INCED(5,11)=8;
INCED(5,13)=6;

INCED(6,1)=9;
INCED(6,2)=9;
INCED(6,6)=7;
INCED(6,7)=0;
INCED(6,8)=8;
INCED(6,9)=8;
INCED(6,11)=0;
INCED(6,13)=8;

INCED(7,1)=5;
INCED(7,4)=7;
INCED(7,5)=9;
INCED(7,6)=7;
INCED(7,7)=7;
INCED(7,9)=8;
INCED(7,11)=8;
INCED(7,12)=9;
INCED(7,13)=8;

INCED(8,4)=8;
INCED(8,5)=6;
INCED(8,6)=8;
INCED(8,7)=9;
INCED(8,9)=8;
INCED(8,11)=8;
INCED(8,12)=9;
INCED(8,13)=9;

INCED(9,1)=5;
INCED(9,5)=8;
INCED(9,6)=9;
INCED(9,8)=4;
INCED(9,9)=5;
INCED(9,10)=3;
INCED(9,11)=8;
INCED(9,13)=7;

INCED(10,5)=6;
INCED(10,6)=0;
INCED(10,7)=8;
INCED(10,8)=9;
INCED(10,9)=5;
INCED(10,11)=6;
INCED(10,13)=8;

INCED(11,5)=8;
INCED(11,6)=9;
INCED(11,8)=9;
INCED(11,9)=8;
INCED(11,11)=6;
INCED(11,13)=7;

INCED(12,1)=8;
INCED(12,6)=8;
INCED(12,8)=9;
INCED(12,13)=8;

INCED(13,6)=5;
INCED(13,7)=8;
```

```
INCED(13,8)=5;
INCED(13,9)=6;
INCED(13,11)=9;
INCED(13,13)=7;

INCED(14,6)=4;
INCED(14,7)=5;
INCED(14,8)=9;
INCED(14,9)=5;
INCED(14,11)=5;
INCED(14,13)=7;

INCED(15,1)=7;
INCED(15,2)=9;
INCED(15,3)=7;
INCED(15,4)=8;
INCED(15,6)=6;
INCED(15,10)=9;
INCED(15,11)=9;
INCED(15,12)=8;
INCED(15,13)=4;

%************************************************************
%                          Fin
%************************************************************
```

```
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
%
%       Fichier:    situation2_DSM.m
%
%


%       Créé par:           G. HARMEL
%                           LAB
%                           Besançon FRANCE
%
%       Date: Juin 2006
%
%
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************

situation2_donnees.m % on appelle les données du problème

%*********************************************
%       traitement flou
%*********************************************

b=newfis('situation 2'); % le traitement flou est appelé situation 2

% Construction des fonctions d'appartenance pour la première entrée
b.input(1).name='SF1-C'; % déclaration du nom de la variable
b.input(1).range=[0 10]; % déclaration du domaine de définition
b.input(1).mf(1).name='Faible'; % nom de la première variable linguistique
b.input(1).mf(1).type='trapmf'; % le type de fonction d'appartenance est trapézoïdale
b.input(1).mf(1).params=[0 0 3 5]; % construction de la fonction trapézoïdale
b.input(1).mf(2).name='Moyen';
b.input(1).mf(2).type='trapmf';
b.input(1).mf(2).params=[3 5 6 8];
b.input(1).mf(3).name='Fort';
b.input(1).mf(3).type='trapmf';
b.input(1).mf(3).params=[6 8 10 10];

% Construction des fonctions d'appartenance pour la deuxième entrée
b.input(2).name='SF2-C';
b.input(2).range=[0 10];
b.input(2).mf(1).name='lFaible';
b.input(2).mf(1).type='trapmf';
b.input(2).mf(1).params=[0 0 3 5];
b.input(2).mf(2).name='Moyen';
b.input(2).mf(2).type='trapmf';
b.input(2).mf(2).params=[3 5 6 8];
b.input(2).mf(3).name='Fort';
b.input(2).mf(3).type='trapmf';
b.input(2).mf(3).params=[6 8 10 10];

% Construction des fonctions d'appartenance pour la sortie
b.output(1).name='SF1-SF2';
b.output(1).range=[0 10];
b.output(1).mf(1).name='Faible';
b.output(1).mf(1).type='trapmf';
b.output(1).mf(1).params=[0 0 1 3];
b.output(1).mf(2).name='Moyen';
b.output(1).mf(2).type='trapmf';
b.output(1).mf(2).params=[1 3 7 9];
b.output(1).mf(3).name='Fort';
```

```
b.output(1).mf(3).type='trapmf';
b.output(1).mf(3).params=[7 9 10 10];

bruleList=[  % définition des règles d'inférence
1 1 1 1 2
3 -1 3 1 1
-1 3 3 1 1
2 2 2 1 1];
b=addrule(b,bruleList);
% fin du traitement flou

%*************************************************************************
%                                application du traitement flou
%*************************************************************************

DSMP = zeros(N_COMP);
DSMF = zeros(N_FS);

for i=1:N_FS
   for j=1:N_COMP
     for k=j+1:N_COMP
      DSMP(i,j,k)=evalfis([INCED(j,i) INCED(k,i)], b); % pour chaque fonction nous construisons une DSM
                                                                                    COMP
       end
   end
end

for i=1:N_COMP
   for j=1:N_FS
     for k=j+1:N_FS
      DSMF(i,j,k)=evalfis([INCED(i,j) INCED(i,k)], b); % pour chaque composant nous construisons une DSM
                                                                                  FS
       end
   end
end

%*************************************************************************
% Filtrage et méthode de la moyenne
%*************************************************************************

DSMPM=zeros(N_COMP); % c'est la DSM résultante issue de l'agrégation par la moyenne et du filtrage
DSMFM=zeros(N_FS);

BASFM=2.35; % définition du seuil de filtrage pour les FS
BASPM=2.0;

    % application à DSMP
for j=1:N_COMP
   for k=j+1:N_COMP
   DSMPM(j,k)=0;
      for i=1:N_FS
        DSMPM(j,k)=DSMPM(j,k)+DSMP(i,j,k);
      end
      if DSMPM(j,k)/N_FS>BASPM;
      DSMPM(j,k)= DSMPM(j,k)/N_FS;
      else
      DSMPM(j,k)=0;
      end
   end
end


% application à DSMFM
for j=1:N_FS
    for k=j+1:N_FS
     DSMFM(j,k)=0;
       for i=1:N_COMP
```

```
        DSMFM(j,k)=DSMFM(j,k)+DSMF(i,j,k);
      end
      if DSMFM(j,k)/N_COMP>BASFM;
       DSMFM(j,k)= DSMFM(j,k)/N_COMP;
    else
      DSMFM(j,k)=0;
     end
   end
end

%************************************
%          symétrie
%************************************
% seule la partie triangulaire supérieure de DSMPM et DSMFM a été construite, nous complétons par symétrie les parties triangulaires inférieures

for j = 1:N_COMP
   for i = j+1:N_COMP
     DSMPM(i,j)=DSMPM(j,i);
   end
end

for j = 1:N_FS
   for i = j+1:N_FS
      DSMFM(i,j)=DSMFM(j,i);
   end
end

%*****************************************************************************
%                                  FIN
%*****************************************************************************
```

La Méthode de coévolution des architectures

Deux fonctions (fichiers) :

- coevo_donnees.m : dans ce fichier, non saisissons les données relatives à la MI Pi-Ai ; à la DSMAi et DSMPi
- coevo_DSM.m : dans ce fichier, nous mettons en œuvre le traitement flou et les autres opérations pour aboutir aux DSM finales DSMAf et DSMPf

```
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
%
% Fichier:          coevo_donnees.m
%
%         Crée PAR:      G. HARMEL
%
%                             LAB
%                             Besançon FRANCE
%
%         Date: Juin 2006
%
%         Initialisation des données pour la méthode de coévolution:
%          les matrices intermédiaires: DSM produit de la BV: DSMPi
%                                       DSM Acteurs: DSMAi
%                                        et la MI: INCED
%
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************

N_P=13; % N_P est le nombre de composants dans la DSM P
N_A=16; % N_A est le nombre d'acteurs dans la DSM A

DSMPi=zeros(N_P,N_P);
DSMAi=zeros(N_A,N_A);
INCED=zeros(N_A,N_P);

%Saisie de la matrice DSMPi
DSMPi(1,2)=7;
DSMPi(1,3)=8;
DSMPi(1,4)=7;
DSMPi(1,5)=9;
DSMPi(1,6)=6;
DSMPi(1,7)=7;
DSMPi(1,8)=9;
DSMPi(1,9)=6;
DSMPi(1,10)=4;
DSMPi(1,11)=7;
DSMPi(1,12)=8;
DSMPi(1,13)=8;

DSMPi(2,3)=7;
DSMPi(2,11)=8;
DSMPi(2,12)=9;

DSMPi(3,4)=7;
DSMPi(3,5)=8;
DSMPi(3,7)=5;
DSMPi(3,8)=5;
DSMPi(3,9)=6;
DSMPi(3,12)=9;
DSMPi(3,13)=9;

DSMPi(4,5)=5;
DSMPi(4,8)=5;
DSMPi(4,9)=9;
DSMPi(4,10)=7;

DSMPi(5,6)=7;
DSMPi(5,7)=7;
```

```
DSMPi(5,8)=8;

DSMPi(6,7)=9;
DSMPi(6,8)=8;
DSMPi(6,13)=6;

DSMPi(8,10)=7;
DSMPi(8,13)=8;

DSMPi(9,10)=8;
DSMPi(9,12)=6;

DSMPi(10,13)=5;

DSMPi(11,12)=7;
DSMPi(11,13)=8;

DSMPi(12,13)=9;

%Saisie de la matrice DSMAi
DSMAi(1,2)=9;
DSMAi(1,3)=7;
DSMAi(1,4)=7;
DSMAi(1,5)=8;
DSMAi(1,6)=6;
DSMAi(1,7)=6;
DSMAi(1,8)=7;
DSMAi(1,9)=6;
DSMAi(1,10)=6;
DSMAi(1,11)=7;
DSMAi(1,12)=6;
DSMAi(1,13)=6;
DSMAi(1,14)=7;
DSMAi(1,15)=6;
DSMAi(1,16)=6;

DSMAi(2,3)=7;
DSMAi(2,4)=7;
DSMAi(2,5)=9;
DSMAi(2,6)=8;
DSMAi(2,7)=8;
DSMAi(2,8)=7;
DSMAi(2,9)=8;
DSMAi(2,10)=8;
DSMAi(2,11)=7;
DSMAi(2,12)=8;
DSMAi(2,13)=8;
DSMAi(2,14)=7;
DSMAi(2,15)=8;
DSMAi(2,16)=8;

DSMAi(3,4)=7;
DSMAi(3,8)=7;
DSMAi(3,11)=6;
DSMAi(3,14)=8;

DSMAi(4,5)=9;
DSMAi(4,6)=9;
DSMAi(4,7)=9;
DSMAi(4,11)=3;
DSMAi(4,14)=7;

DSMAi(5,7)=9;
DSMAi(5,11)=5;
DSM Ai(5,13)=5;

DSMAi(8,9)=9;
```

```
DSMAi(8,10)=9;

DSMAi(11,12)=9;
DSMAi(11,13)=9;
DSMAi(11,14)=7;

DSMAi(12,13)=8;

DSMAi(14,15)=8;
DSMAi(14,16)=8;

DSMAi(15,16)=9;

%Saisie de la MI INCED
INCED(1,1)=5;
INCED(1,2)=5;
INCED(1,3)=5;
INCED(1,4)=5;
INCED(1,5)=5;
INCED(1,6)=5;
INCED(1,7)=5;
INCED(1,8)=9;
INCED(1,9)=5;
INCED(1,10)=5;
INCED(1,11)=5;
INCED(1,12)=5;
INCED(1,13)=5;


INCED(2,1)=5;
INCED(2,2)=5;
INCED(2,3)=5;
INCED(2,4)=5;
INCED(2,5)=5;
INCED(2,6)=6;
INCED(2,7)=6;
INCED(2,8)=9;
INCED(2,9)=6;
INCED(2,10)=6;
INCED(2,11)=6;
INCED(2,12)=6;
INCED(2,13)=6;

INCED(3,1)=9;
INCED(3,3)=7;
INCED(3,5)=7;

INCED(4,1)=8;
INCED(4,3)=5;
INCED(4,5)=9;
INCED(4,6)=8;
INCED(4,7)=8;
INCED(4,8)=9;

INCED(5,4)=5;
INCED(5,5)=9;
INCED(5,6)=5;
INCED(5,8)=9;
INCED(5,10)=7;
INCED(5,13)=5;

INCED(6,5)=8;
INCED(6,7)=9;
INCED(6,6)=5;

INCED(7,5)=8;
INCED(7,6)=9;
```

```
INCED(7,7)=5;
INCED(7,8)=5;

INCED(8,1)=7;
INCED(8,2)=9;
INCED(8,11)=7;
INCED(8,12)=7;

INCED(9,2)=8;
INCED(9,11)=9;
INCED(9,12)=5;

INCED(10,2)=8;
INCED(10,11)=5;
INCED(10,12)=9;
INCED(10,13)=5;

INCED(11,1)=7;
INCED(11,4)=9;
INCED(11,8)=3;
INCED(11,9)=7;
INCED(11,10)=7;

INCED(12,4)=7;
INCED(12,9)=9;
INCED(12,10)=5;

INCED(13,4)=7;
INCED(13,8)=5;
INCED(13,9)=5;
INCED(13,10)=9;

INCED(14,1)=7;
INCED(14,2)=5;
INCED(14,3)=9;
INCED(14,13)=9;

INCED(15,3)=8;
INCED(15,13)=9;

INCED(16,3)=8;
INCED(16,12)=8;
INCED(16,13)=8;

%*********************************************************************
% Symétrie
%*********************************************************************

%% dans la partie saisie, nous avons considéré seulement les parties
%% triangulaires supérieures. Dans cette partie on construit par symétrie les
%% parties triangulaires inférieures

for j = 1:N_P
   for i = j+1:N_P
      DSMPi(i,j)=DSMPi(j,i);
   end
end

for j = 1:N_A
   for i = j+1:N_A
      DSMAi(i,j)=DSMAi(j,i);
   end
end

%initialisation de la diagonale à 10
for i=1:N_A
   DSMAi(i,i)=10;
```

```
end
for i=1:N_P
   DSMPi(i,i)=10;
end

%*************************************************************************
%           FIN
%*************************************************************************
```

```
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
%
%       File:       coevo_DSM.m
%
%
%       Créé par : G. HARMEL
%                   LAB
%                   Besançon FRANCE
%
%       Date: Juin 2006
%
%       Ce fichier contient le traitement flou permettant de générer les DSM
%       coévoluées et finales
%
%
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************

coevo_donnees.m % on appelle les données du problème

%*********************************************************************
%       PROCESSUS FLOUS
%*********************************************************************



fuzzy_co=newfis('COEVOLUTION'); % Le TRAITEMENT FLOU est appelé COEVOLUTION

% Construction des fonctions d'appartenance pour la première entrée
fuzzy_co.input(1).name='Ai-Aj';    % déclaration du nom de la variable
fuzzy_co.input(1).range=[0 10];    % déclaration du domaine de définition
fuzzy_co.input(1).mf(1).name='FAIBLE';    % nom de la première variable linguistique
fuzzy_co.input(1).mf(1).type='trapmf';    % le type de fonction d'appartenance est trapézoïdale
fuzzy_co.input(1).mf(1).params=[0 0 3 5];    % construction de la fonction trapézoïdale
fuzzy_co.input(1).mf(2).name='MOYEN';
fuzzy_co.input(1).mf(2).type='trapmf';
fuzzy_co.input(1).mf(2).params=[3 5 6 8];
fuzzy_co.input(1).mf(3).name='FORT';
fuzzy_co.input(1).mf(3).type='trapmf';
fuzzy_co.input(1).mf(3).params=[6 8 10 10];

% Construction des fonctions d'appartenance pour la deuxième entrée
fuzzy_co.input(2).name='Ai-Bk';
fuzzy_co.input(2).range=[0 10];
fuzzy_co.input(2).mf(1).name='FAIBLE';
fuzzy_co.inp ut(2).mf(1).type='trapmf';
fuzzy_co.input(2).mf(1).params=[0 0 3 5];
fuzzy_co.input(2).mf(2).name='MOYEN';
fuzzy_co.input(2).mf(2).type='trapmf';
fuzzy_co.input(2).mf(2).params=[3 5 6 8];
fuzzy_co.input(2).mf(3).name='FORT';
fuzzy_co.input(2).mf(3).type='trapmf';
fuzzy_co.input(2).mf(3).params=[6 8 10 10];

% Construction des fonctions d'appartenance pour la troisième entrée
fuzzy_co.input(3).name='Aj-Bv';
fuzzy_co.input(3).range=[0 10];
fuzzy_co.input(3).mf(1).name='FAIBLE';
fuzzy_co.input(3).mf(1).type='trapmf';
fuzzy_co.input(3).mf(1).params=[0 0 3 5];
fuzzy_co.input(3).mf(2).name='MOYEN';
fuzzy_co.input(3).mf(2).type='trapmf';
fuzzy_co.input(3).mf(2).params=[3 5 6 8];
```

```
fuzzy_co.input(3).mf(3).name='FORT';
fuzzy_co.input(3).mf(3).type='trapmf';
fuzzy_co.input(3).mf(3).params=[6 8 10 10];
% Construction des fonctions d'appartenance pour la sortie
fuzzy_co.output(1).name='Bu-Bv';
fuzzy_co.output(1).range=[0 10];
fuzzy_co.output(1).mf(1).name='FAIBLE';
fuzzy_co.output(1).mf(1).type='trapmf';
fuzzy_co.output(1).mf(1).params=[0 0 1 3];
fuzzy_co.output(1).mf(2).name='MOYEN';
fuzzy_co.output(1).mf(2).type='trapmf';
fuzzy_co.output(1).mf(2).params=[1 3 7 9];
fuzzy_co.output(1).mf(3).name='FORT';
fuzzy_co.output(1).mf(3).type='trapmf';
fuzzy_co.output(1).mf(3).params=[7 9 10 10];

fuzruleList=[ % définition des 13 règles d'inférence
1 0 0 1 1 1
2 1 0 1 1 1
2 0 1 1 1 1
2 2 2 2 1 1
2 3 2 3 1 1
2 2 3 3 1 1
2 3 3 3 1 1
3 1 1 1 1 1
3 1 -1 2 1 1
3 -1 1 2 1 1
3 2 -1 3 1 1
3 -1 2 3 1 1
3 3 3 3 1 1 ];

fuzzy_co=addrule(fuzzy_co,fuzruleList);
% fin du traitement flou

%*************************************************************************
%   application du traitement flou
%*************************************************************************
DSMP=zeros(N_A,N_A,N_P,N_P);
DSMA=zeros(N_P,N_P,N_A,N_A);

for i=1:N_A
   for j=1:N_A
     for k=1:N_P
        for u=k+1:N_P
          DSMP(i,j,k,u)=evalfis([DSMAi(i,j) INCED(i,k) INCED(j,u)], fuzzy_co);
          % pour chaque interaction dans DSMAi (i,j) nous construisons
          % une DSM du produit à travers le traitement flou
          end
       end
   end
end

for i=1:N_P
   for j=1:N_P
     for k=1:N_A
        for u=k+1:N_A
          DSMA(i,j,k,u)=evalfis([DSMPi(i,j) INCED(k,i) INCED(u,j)], fuzzy_co);
          % pour chaque interaction dans DSMPi (i,j) nous construisons
          % une DSM des acteurs à travers le traitement flou
          end
       end
   end
end

%*************************************************************************
% construction des DSM coévoluées
%*************************************************************************
```

```
% utilisation de la méthode de la moyenne
DSMPc=zeros(N_P,N_P);
DSMAc=zeros(N_A,N_A);


for k=1:N_P
   for u=k+1:N_P
      DSMPc(k,u)=0;
      for i=1:N_A
         for j=1:N_A
            DSMPc(k,u)=DSMPc(k,u)+DSMP(i,j,k,u);
         end
      end
   end
end

DSMPc=DSMPc/(N_A^2);

for k=1:N_A
   for u=k+1:N_A
      DSMAc(k,u)=0;
      for i=1:N_P
         for j=1:N_P
            DSMAc(k,u)=DSMAc(k,u)+DSMA(i,j,k,u);
         end
      end
   end
end

DSMAc=DSMAc/(N_P^2);

% Normalisation des DSM coévoluées
DSMPc = DSMPc/max(max(DSMPc))*10;
DSMAc = DSMAc/max(max(DSMAc))*10;


%filtrage des DSM coévoluées

BASA=2.3;
BASP=2.4;

for j=1:N_P
   for k=j+1:N_P
      if DSMPc(j,k)<BASP
      DSMPP(j,k)=0;
      end
   end
end

for j=1:N_A
   for k=j+1:N_A
      if DSMAc(j,k)<BASA
      DSMAc(j,k)=0;

      end
   end
end

%*************************************************************************
%CONSTRUCTION DES DSM FINALES
%*********************************************************************
DSMPf=zeros(N_P,N_P);
DSMAf=zeros(N_A,N_A);

DSMPf= 0.5*DSMPc + 0.5*DSMPi;
DSMAf= 0.5*DSMAc + 0.5*DSMAi;
```

```
%filtrage des DSM finales

BASAF=2.3;
BASPF=2.4;

for j=1:N_P
   for k=j+1:N_P
      if DSMPf(j,k)<BASPF
      DSMPf(j,k)=0;
      end
   end
end

for j=1:N_A
   for k=j+1:N_A
      if DSMAf(j,k)<BASAF
      DSMAf(j,k)=0;

      end
   end
end


%*************************************************************************
%            SYMETRIE
%*************************************************************************

% on a construit la partie triangulaire supérieure et nous complétons dans
% cette partie la symétrie des DSM
for j = 1:N_P
   for i = j+1:N_P
      DSMPc(i,j)=DSMPc(j,i);
      DSMPf(i,j)=DSMPf(j,i);
   end
end

for j = 1:N_A
   for i = j+1:N_A
      DSMAc(i,j)=DSMAc(j,i);
      DSMAf(i,j)=DSMAf(j,i);
   end
end
%*************************************************************************
%*************************************************************************
%             FIN
%*************************************************************************
%*************************************************************************
```

L’algorithme de clustering

Dans cette partie, nous présentons les fonctions qu’on a développées.

Algorithme initial	Notre algorithme
run_cluster	Architecture_DSM
bid	enchere
coord_cost	Cout_Couplage

La fonction architecture_DSM sera illustrée sur la DSMPM issue de situation2_DSM.m

```
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************
%
%       fichier:    Architecture_DSM.m
%
%       Crée par: G. HARMEL
%                   LAB
%                   Besançon FRANCE
%
%       Date: Juin 2006
%
% ********************************************************************************
% ********************************************************************************
% ********************************************************************************


% ********************************************************************************
%                               appeler la fonction qui donne la DSM résultante
% exemple : situation2_DSM.m
% ********************************************************************************


% ********************************************************************************
%                                   Initialisation des PARAMETRES
                                                    *
% ********************************************************************************
%       print_flag                  quand à 1 donne autorisation d'imprimer
%       extract_elements            permet à l'utilisateur d'identifier directement des éléments intégrateurs

%                                                                               *
% ********************************************************************************

print_flag = 0;
extract_elements = [];
IC=0.8;                 % choisir une valeur pour l'Indice de Couplage
% les paramètres sont expliqués dans le paragraphe 3.5 du chapitre III
Cluster_param.exp_int           =  2;
Cluster_param.exp_taille        =  1;
Cluster_param.rand_accept       =  15;
Cluster_param.rand_bid          =  15;
Cluster_param.times             =  4;
Cluster_param.stable_limit      =  4;

% ********************************************************************************
% **********************  Fin saisie des paramètres*******************************
% ********************************************************************************



% ********************************************************************************
%       extraction des données
% ********************************************************************************

DSM_matrix = DSMPM ; % on spécifie la matrice à utiliser
DSM_matrix_original = DSM_matrix;
%************************************************
%       identification des intégrateurs par IC
%************************************************

if (IC>0) % si IC=0 alors on n'utilise pas la procédure

   for i=1:DSM_size
      inc2=0;
      inc=0;
```

```
      ICD(i)=0;
      for j=1:DSM_size
         if (DSM_matrix(i,j)>0)
            inc=inc+1;
         end
      end
      ICD(i)=inc/(DSM_size-1);
      if (ICD(i)>=IC)
         inc2=inc2+1;
         extract_elements(length(extract_elements)+inc2)=i;
      end
   end
end
extract_elements

% initialiser les paramètres à zéro
for i = 1: length(extract_elements)
  DSM_matrix(extract_elements(i),:) = 0;
  DSM_matrix(:,extract_elements(i)) = 0;
end
% ******************  Fin extraction des données**************************


% ********************************************************************************
%                                  commencer le CLUSTERING
% ********************************************************************************

[Cluster_matrix, cout_total_couplage, historique_cout, old_data] = Cluster(DSM_matrix, Cluster_param);

% Cluster est la fonction principale qui appelle toutes les autres fonctions réalisant le clustering
% ************************** Fin CLUSTERING ********************************


% ********************************************************************************
% ********************************************************************************
%
%                                                          Gestion des graphiques
% ********************************************************************************
% ********************************************************************************

% appeler l'historique des coûts
[cost_g_zero, cg] = find(historique_cout);
max_run = max(cost_g_zero);

% trier les matrices par la taille
[Cluster_matrix] = reorder_cluster(Cluster_matrix);  % sort cluster by cluster size

% Placer 1 sur la diagonale
[graph_DSM_matrix] = place_diag(DSM_matrix_original, 1);
[graph_New_DSM_matrix] = place_diag(New_DSM_matrix, 1);

get_date = now;
current_date = datestr(get_date,0);

% Créer les titres des graphes
DSM_title = ['DSM ;  ' ];
Cluster_title = ['Cluster Matrix;  '];
New_DSM_title = ['DSM ; ' '  Coût Total : ' num2str(cout_total_couplage)];

% dessiner la matrice originale
graph_matrix(graph_DSM_matrix,'Element','Element',DSM_title, DSM_labels, DSM_labels, print_flag);


% dessiner l'historique de cout
figure;
plot(historique_cout(1:max_run));
title(['Clustering Historique des Coûts;  ' current_date]);
```

```
xlabel('Change #');
ylabel('Cost');
orient landscape;
if print_flag ==1
  print;
end


%  afficher la DSM avec l'architecture
graph_matrix(graph_New_DSM_matrix,'Element','Element',New_DSM_title,New_DSM_labels,          New_DSM_labels,
print_flag, Cluster_matrix);

% ************************************************************************
%                                          Fin graphique

%************************************************************************
```

```
% ********************************************************************************
% ********************************************************************************
%
%  Fichier :           enchere.m
%
%           Crée par:       G. HARMEL
%                                   LAB
%                                   Besançon FRANCE
%           Date: Juin 2006
%
%
%
%
% ********************************************************************************
% ********************************************************************************
% *
function [cluster_enchere]=enchere(elmt, DSM_matrix, cluster_matrix, cluster_size,exp_int,exp_taille);
%
%
% Fonction qui calcule l'enchère de tous les modules vers un élément elmt
%
%
%
%           Données:
%                   elmt                    élément qui recevra les enchères des modules
%                   DSM_matrix              la DSM analysée
%                   Cluster_matrix          matrice (cluster,element)
%                                           1 = si élément dans cluster, 0 = sinon
%                   cluster_size            Tableau des tailles des clusters
%                   exp_int                  pour favoriser les interactions
%                   exp_taille              pénalité associée à la taille des modules
%
%           Sortie:
%                   cluster_enchere         tableau des enchères
%********************************************************************************
%********************************************************************************

%   appeler le nombre de clusters et le nombre d'éléments dans la DSM
[n_clusters, DSM_size] = size(cluster_matrix);

%   intialisation du tableau d'enchere
cluster_enchere = zeros(n_clusters,1);
CM(index)= zeros(n_clusters,1);

%   on calcule la cohésion des modules déjà existants
for (index=1:n_clusters)
   in(index)=0;
      if (cluster_size(index)>1)
      for i=1:DSM_size
         if (Cluster_matrix(index,i)==1)
            for j=i+1:DSM_size
               if (Cluster_matrix(index,j)==1)
                  if(DSM_matrix(i,j)>0)
                     in(index) = in(index) + (DSM_matrix(i,j) + DSM_matrix(j,i));
                  end
               end
            end
         end
      end
      CM(index)=((in(index))^exp_int)/(cluster_size(i)^exp_taille); %Eq. III-
   end
end
CM_elmt=zeros(n_clusters,1);  % CM_elment est CM(cluster+ elmt) la cohésion des modules si on leur associe elmt

for index=1:n_clusters
   if (cluster_matrix(index,elmt)==1)
      cluster_enchere(index)=0;    % si elmt est dans le cluster, l'enchère est nulle
```

```
   else                        % on considère le cas où elmt n'appartient pas à cluster
     inter_elmt(index)=0;          % cette variable contient la somme des interactions entre le cluster et elmt
      for j=1:DSM_size
         if((cluster_matrix(index,j)==1)&(j~=elmt))
            if ((DSM_matrix(j,elmt)>0))
               inter_elmt(index) = inter_elmt(index) + DSM_matrix(j,elmt)+ DSM_matrix(elmt,j);
            end
         end
      end
      CM_elmt(index)=((in(index)+inter_elmt(index))^exp_int)/((cluster_size(i)+1)^exp_taille);
   end
end
cluster_enchere=CM_elmt-CM;

%*********************************************************************
%*********************************************************************
%                        Fin enchere.m
%*********************************************************************
%*********************************************************************
```

```
function [cout_total_couplage] = Cout_Couplage(DSM_matrix, Cluster_matrix, cluster_size);
%
%
%                   Données:
%                                   DSM_matrix          La DSM analysée
%                                   Cluster_matrix      la matrice des modules(Cluster,Element)
%                                   cluster_size(n)     Tableau des tailles des clusters
%
%
%                   Sortie:
%                                   cout_total_couplage     c'est le coût de la fonction objectif
%
%   cette fonction classe les interactions en interactions internes ou
%   externes aux modules. Pour chaque classe, nous associons un coût de
%   couplage. Ce coût de couplage est ensuite corrigé par un indicateur MSI.
%

% ************************************************************************
% ************************************************************************
% ************************************************************************
%   File:  cout_couplage.m
%          Crée par:      G. HARMEL
%                                          LAB
%                                          Besançon FRANCE
%          Date: Juin 2006
%
% ************************************************************************
% ************************************************************************
% ************************************************************************

%   appeler le nombre de clusters et le nombre d'éléments dans la DSM
[n_clusters, DSM_size] = size(Cluster_matrix);

% intialiser le cout total
cout_total_couplage = 0;
cout_couplage= zeros(1,DSM_size);

%   On réordonne la DSM selon la cluster_matrix
%   REMARQUE: ce réarrangement va dupliquer les éléments qui appartiennent
%   à plusieurs modules.
%   si un élément appartient à trois modules, la nouvelle DSM aura trois
%   entrées différentes pour cet élément. Le coût associé à une telle architecture augmente
%   avec cette duplication.

[New_DSM_matrix, New_DSM_labels] = reorder_DSM_byCluster(DSM_matrix, Cluster_matrix, DSM_labels);
New_DSM_size = size(New_DSM_matrix,1);

% capter le nombre d'élément dans chaque module
Num_cluster_elements = sum(Cluster_matrix,2);

n=1;

New_Cluster_matrix = zeros(New_DSM_size, New_DSM_size);


for i =1:n_clusters
  New_Cluster_matrix(i,n:n+Num_cluster_elements(i)-1) = ones(1,Num_cluster_elements(i));
  n= n+Num_cluster_elements(i);
end

% obtenir le tableau “new cluster size” qui correspond à la nouvelle matrice new cluster matrix
New_Cluster_size = sum(New_Cluster_matrix,2);

% remplacer les données anciennes avec les nouvelles données pour le calcul
% des coûts

DSM_size = New_DSM_size;
```

```
DSM_matrix = New_DSM_matrix;
Cluster_matrix = New_Cluster_matrix;
cluster_size = New_Cluster_size;

[n_clusters, DSM_size] = size(Cluster_matrix);
cout_total_couplage = 0;



% ************************************************************************
% ************************************************************************
%                          CALCUL DU COUT DE L'ARCHITECTURE PROPOSEE
% ************************************************************************
% ************************************************************************

%*****************************************
% Calcul des couts de couplage internes aux modules
%*****************************************************

% la déclaration des variables n'est pas nécessaire mais cela aide à
% contrôler le bon fonctionnement de l'algorithme

cout_in=zeros(n_clusters,1);
MSIi=zeros(n_clusters,1);
cout_in_corrig=zeros(n_clusters,1);
cout_total_in_corrig=0;

for (index=1:n_clusters)
   somme(index)=0;
   inc=0;
   if (cluster_size(index)>1)
      for i=1:DSM_size
         for j=i+1:DSM_size
            if (Cluster_matrix(index,i)==1)
               if (Cluster_matrix(index,j)==1)
                  if(DSM_matrix(i,j)>0)
                  somme(index) = somme(index) + (DSM_matrix(i,j) + DSM_matrix(j,i));
                  inc=inc+2;
                  end
               end
            end
         end
      end
      cout_in(index)=somme(index)*((cluster_size(index))^2);  % Eq. III-7
      MSIi(index)=inc/((cluster_size(index))^2-cluster_size(index));  % Eq. III-13
      cout_in_corrig(index)=cout_in(index)/MSIi(index);  % Eq. III-15
   else
      cout_in_corrig(index)=0;
   end
end

cout_total_in_corrig = sum(cout_in_corrig);% Eq. III-17

%*****************************************************************
% Calcul des coûts de couplage internes aux modules
%*****************************************************************

cout_out=zeros(n_clusters,n_clusters);
MSIe=zeros(n_clusters,n_clusters);
cout_out_corrig=zeros(n_clusters,n_clusters);
cout_total_out_corrig=0;

for (index=1:n_clusters)
   for (index2=index+1:n_clusters)
      inc2=0;
      for i=1:DSM_size
         if (Cluster_matrix(index,i)==1)
```

```
            for j=i:DSM_size
               if (Cluster_matrix(index2,j)==1)
                  if(DSM_matrix(i,j)>0)
                     inc2=inc2+2;
                     cout_out(index,index2)=cout_out(index,index2)+
                     (DSM_matrix(i,j)+DSM_matrix(j,i))*(DSM_size + cluster_size(index) + cluster_size(index2))^2;
                     % Eq. III-8
                  end
               end
            end
         end
      end
      if (cluster_size(index)==0|cluster_size(index2)==0)
         if (inc2==0)
            MSIe(index,index2) =1;
         end
      else
         if (inc2==0)
            MSIe(index,index2) =1;
         else
            MSIe(index,index2) = inc2/(2*cluster_size(index)*cluster_size(index2)); % Eq. III-14
         end
      end
      cout_out_corrig(index,index2)=cout_out(index,index2)*MSIe(index,index2); % Eq. III-16
   end
end

cout_total_out_corrig=sum(sum(cout_out_corrig));% Eq. III-17

%*****************************************************************
%                        CALCUL DU COUT TOTAL
%*****************************************************************
cout_total_couplage = cout_total_in_corrig + cout_total_out_corrig;  % Eq. III-17


% ***************************************************************************
% ******************** FIN du calcul du coût ********************************
% ***************************************************************************
```

Vers une conception conjointe des architectures du produit et de l'organisation du projet dans le cadre de l'Ingénierie Système

Résumé

Lorsqu'une entreprise prend la décision stratégique de lancer une nouvelle famille de produits ou de reconcevoir un produit existant, l'architecte système a pour mission de concevoir ou de faire évoluer l'architecture de ce produit. L'architecte joue aussi le rôle de chef de projet et doit concevoir ou faire évoluer en même temps, l'organisation du projet pour la rendre plus performante. Dans ce mémoire, notre objectif est de développer des modèles et méthodes permettant d'aider les architectes système dans cette double activité.

Dans le cadre de l'Ingénierie Système, notre méthode se base sur la définition de nos propres concepts d'architecture et de conception modulaire pour les étendre à la définition de l'architecture de l'organisation du projet. Nous proposons ensuite en cohérence avec notre positionnement, un algorithme de clustering utilisant l'outil DSM comme méthode de représentation, cet algorithme a pour fonction de révéler l'architecture d'un domaine en partant de sa représentation matricielle (DSM).

L'application de notre méthode de développement des architectures est liée aux quatre situations de conception identifiées. Pour chacune de ces situations, nous proposons une méthode de conception des architectures, faisant appel à un traitement flou et/ou à des opérations matricielles. Chacune de ces situations est ensuite illustrée par une application à la conception d'un moteur thermique dans l'industrie automobile. La démarche présentée dans ce chapitre est une vision statique de la conception des architectures.

Face cette vision statique, nous montrons la nécessité de faire « coévoluer » les architectures couplées. Nous proposons alors l'exploration des incertitudes comme méthode pour suivre l'évolution des systèmes (perturbations) étudiés. Nous développons une méthode basée sur un traitement flou pour faire coévoluer les architectures perturbées et pour les rendre cohérentes.

Mots-clés : Architecture, conception modulaire, DSM, logique floue, algorithmes de clustering, gestion des incertitudes.

Identifying Product and Organization architectures in the context of System Engineering

Abstract

When a company makes the strategic decision of launching a new product family or redesigning an existing product, the system architect is in charge of designing and making evolve the architecture of this product. The system architect is also the Development Project (DP) manager and thus he is in charge of conceiving or making evolve at the same time the DP organization (Design Teams).

In this memory, our objective was to develop models and methods making it possible to help the system architects in this double duty. In short, we propose in this work a method for designing both product and organization architectures in the preliminary phases of the DP. In Engineering System context, our method is based on the definition of our own concepts of product architecture and modularity and in their extension to DP organization. We propose then in coherence with our positioning, a clustering algorithm using DSM representation tool, the algorithm objective is to reveal project domains architectures starting from their matrix representation (DSM).

The use of our method is tightly related to the design situations that the architect can face. In this memory, we identified four of them. For each one of these situations, we propose a method for architectures designing, calling upon a Fuzzy Logic treatment. Each one of these situations is then illustrated by an application to the design of a thermal engine in the car industry. The step presented in this chapter is a static vision of the design of architectures.

By opposition to this static vision, we show the need for making co-evoluate coupled architectures. We propose then the exploration of uncertainties to model systems evolution. We develop a method based on a Fuzzy treatment to make co-evoluate disturbed architectures in order to obtain coherent ones.

Keywords: Product architecture, modular design, Design Structure Matrix, Fuzzy Logic, Clustering algorithm, uncertainty management.